

# DB2 Content Manager for z/OS V8.3

## Implementation, Installation, and Migration

Toolkit, host integration, security,  
exits, and sample source code

Migration from ImagePlus, CM  
V2.3 (OS/390), and CM V7 (MP)

Advanced configuration  
topics



Wei-Dong Zhu  
Claus Heisterberg-Andersen  
Marcelo Takashi Souza  
Jaco van Straaten  
Rob Weaver





International Technical Support Organization

**DB2 Content Manager for z/OS V8.3:  
Implementation, Installation, and Migration**

August 2005

Archived

**Note:** Before using this information and the product it supports, read the information in “Notices” on page xv.

### **First Edition (August 2005)**

This edition applies to Version 8, Release 3, of IBM DB2 Content Manager for z/OS (product number 5697-H60).

**© Copyright International Business Machines Corporation 2005. All rights reserved.**

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

# Contents

<b>Figures</b> .....	xiii
<b>Notices</b> .....	xv
Trademarks .....	xvi
<b>Preface</b> .....	xvii
The team that wrote this IBM Redbook .....	xvii
Become a published author .....	xix
Comments welcome .....	xix
<b>Part 1. Introduction and implementation</b> .....	1
<b>Chapter 1. Introduction</b> .....	3
1.1 Background .....	4
1.2 What does Content Manager do? .....	4
1.3 Content Manager system architecture overview .....	5
1.4 Enhancements in Content Manager for z/OS V8.3 .....	6
1.5 Differences between ImagePlus OS/390 and CM V8.3 .....	8
1.6 Differences between CM for z/OS and Multiplatforms .....	10
<b>Chapter 2. Implementation</b> .....	13
2.1 Analyze business requirements .....	14
2.2 Plan system topology .....	15
2.2.1 DB2 database .....	16
2.2.2 Library Server .....	16
2.2.3 Resource Manager .....	16
2.2.4 Storage subsystem .....	17
2.2.5 HTTP server .....	17
2.2.6 System administration client .....	18
2.2.7 Windows client (Client for Windows) .....	18
2.2.8 Web interface .....	18
2.2.9 Content Manager Toolkit for z/OS .....	19
2.3 Define data model .....	19
2.3.1 Plan your data model .....	20
2.3.2 Data model entities .....	22
2.4 Implement user access .....	32
2.4.1 Users and user groups .....	33
2.4.2 Privileges, privilege groups, and privilege sets .....	33
2.4.3 Access control lists .....	34

2.5	Implement workflow	34
2.5.1	Actions and action lists	35
2.5.2	Workflow nodes	36
2.5.3	Workflow processes	36
2.5.4	Work lists	37
2.5.5	Work packages	37
2.6	Manage object storage	38
2.6.1	Storage management systems	38
2.6.2	Collections	38
2.6.3	Capacity planning	39
2.6.4	Migration policies	39
2.6.5	Replication	40
2.7	Manage different domains	40
2.8	Provide Web access	41
2.9	Backup and recovery	42
2.10	Performance tuning	42
2.11	Additional products that use Content Manager	43
<b>Chapter 3.</b>	<b>Content Manager Toolkit for z/OS</b>	<b>45</b>
3.1	Introduction	46
3.2	Configuring the z/OS development environment	46
3.2.1	Configuring environment variables	46
3.2.2	Executing Content Manager Toolkit's environment script	47
3.3	Content Manager Toolkit for z/OS interfaces overview	47
3.3.1	Java interface	48
3.3.2	Non-visual bean interface	48
3.3.3	Visual bean interface	48
3.3.4	Content Manager Toolkit for z/OS package imports	48
3.3.5	Error or exception handling	49
3.4	Compiling and running the sample programs	50
3.4.1	Compiling and executing ICM sample programs	50
3.4.2	Compiling and executing non-visual bean samples	52
3.5	Sample ICM programs provided on z/OS	53
3.5.1	Preparing the ICM sample programs	54
3.5.2	Transaction sample: STransactionsICM.java	56
3.5.3	Item creation sample: SItemCreationICM.java	56
3.5.4	Item deletion sample: SItemDeletionICM.java	56
3.5.5	Item retrieval sample: SItemRetrievalICM.java	56
3.5.6	Item modification sample: SItemUpdateICM.java	56
3.5.7	Folder manipulation sample: SFolderICM.java	57
3.5.8	Link manipulation sample: SLinksICM.java	57
3.5.9	Search sample: SSearchICM.java	57
3.5.10	Document routing sample: SDocRoutingListingICM.java	58

3.5.11 Deleting sample data . . . . .	58
3.6 Sample Java bean programs provided on z/OS . . . . .	58
3.6.1 Link item sample: TAddLink.java . . . . .	59
3.6.2 Folder creation sample: TCreateFolder.java . . . . .	59
3.6.3 Item creation sample: TCreateItem.java . . . . .	59
3.6.4 Deletion sample: TDelete.java . . . . .	60
3.6.5 Attribute update sample: TEditAttributes.java . . . . .	60
3.6.6 Entity listing sample: TListEntities.java . . . . .	60
3.6.7 Link listing sample: TListLinks.java . . . . .	61
3.6.8 Search sample: TSearch.java . . . . .	61
3.6.9 Link removal sample: TRemoveLink.java . . . . .	61
3.7 Using Content Manager Toolkit for z/OS from a batch environment. . . . .	62
3.7.1 Setting up data and a data model for the batch process. . . . .	62
3.7.2 Setting up JCL for batch process . . . . .	63
3.7.3 Developing business logic in a Java program. . . . .	64
3.7.4 Calling a Java program . . . . .	67
3.8 Using Content Manager Toolkit for z/OS from a Web environment . . . . .	68
3.8.1 Using WebSphere . . . . .	76
<b>Chapter 4. Host integration . . . . .</b>	<b>79</b>
4.1 Introduction . . . . .	80
4.2 Host integration using Content Manager Toolkit for z/OS . . . . .	80
4.3 Document routing user exit routines . . . . .	80
4.3.1 When to use document routing exits. . . . .	81
4.3.2 Sample document routing user exits. . . . .	82
4.3.3 Development considerations. . . . .	83
4.3.4 Document routing user exit header file . . . . .	83
4.3.5 Document routing container data . . . . .	84
4.3.6 Document routing user exit interface . . . . .	85
4.3.7 Document routing user exit function prototype . . . . .	87
4.3.8 Document routing user exit return code . . . . .	87
4.3.9 Sample document routing user exit. . . . .	87
4.3.10 How to compile and link the sample user exits . . . . .	89
4.3.11 Configure document routing user exit . . . . .	93
4.3.12 Using document routing user exits . . . . .	96
4.4 Host integration using database triggers. . . . .	97
4.5 Host integration using security exits . . . . .	97
4.6 Extend order received by Resource Manager. . . . .	97
4.6.1 When to use Resource Manager user exits . . . . .	98
4.6.2 Predefined user exit names . . . . .	98
4.6.3 Resource Manager user exit interface . . . . .	100
4.6.4 Resource Manager user exit function prototype . . . . .	100
4.6.5 Example of Resource Manager user exit . . . . .	100

4.6.6	How to use Resource Manager user exits .....	103
4.6.7	Dealing with large Resource Manager collections .....	104
4.7	Integration from the Windows client .....	106
4.7.1	A practical example .....	106
4.7.2	Available Windows client user exits .....	107
4.8	Integration from eClient .....	108
4.9	Integration from Information Integrator for Content environment .....	109
<b>Chapter 5.</b>	<b>Security and exits .....</b>	<b>111</b>
5.1	Security in Content Manager overview .....	112
5.2	RACF logon exit implementation .....	112
5.2.1	Activating RACF exit .....	114
5.2.2	APF authorizing Content Manager load library .....	114
5.2.3	Using security managers other than RACF .....	114
5.2.4	Users not validated by the security exit .....	115
5.2.5	Importing user definitions from RACF .....	115
5.2.6	Specifying ACLs and privilege sets for imported users .....	116
5.2.7	Sample SQLs to extract information from Content Manager .....	119
5.2.8	Extract user list and information from Content Manager .....	123
5.2.9	Alternative ways of providing input to user import job .....	124
5.2.10	Copy encrypted password from sample user .....	124
5.3	ACL user exit routines overview .....	125
5.4	ICMACLPrivExit .....	126
5.4.1	ICMACLPrivExit definition in DB2 .....	126
5.4.2	Activating ICMACLPrivExit .....	127
5.4.3	Sample ACL exit program and parameter list .....	128
5.4.4	Call types .....	130
5.5	ICMGenPrivExit .....	131
5.5.1	Activating ICMGenPrivExit .....	132
5.5.2	Sample exit program and parameter list .....	132
5.6	Writing exits in non-C programming languages .....	132
5.6.1	Redefining UDF .....	133
5.6.2	Sample PL/I program with parameter list .....	134
5.6.3	Linking exit program .....	137
5.7	Additional options with DB2 triggers .....	138
5.7.1	DB2 trigger overview .....	138
5.7.2	Using DB2 triggers with Content Manager .....	138
5.7.3	Example of a simple trigger calling a UDF .....	139
5.7.4	Calling a program from a trigger .....	139
5.7.5	Update Content Manager to handle SQL return codes .....	143
5.8	Resource Manager exits .....	143
5.8.1	Parameters passed to Resource Manager exit programs .....	144
5.8.2	Activating Resource Manager exits .....	145



5.8.3 Making exit programs available to Resource Manager . . . . .	146
5.8.4 What can you do in Resource Manager exits . . . . .	146
5.8.5 Illustrating the use of the PRE- and POSTSTORE exit. . . . .	147
<b>Part 2. Installation and migration . . . . .</b>	<b>151</b>
<b>Chapter 6. Installation . . . . .</b>	<b>153</b>
6.1 Content Manager installation and configuration . . . . .	154
6.1.1 Planning for installation . . . . .	154
6.1.2 Library Server installation . . . . .	157
6.1.3 Resource Manager installation . . . . .	160
6.1.4 Post installation. . . . .	161
6.2 Content Manager Toolkit for z/OS V8.3 installation . . . . .	161
6.2.1 Configuring Toolkit . . . . .	163
6.2.2 Configuring Content Manager Toolkit for DB2 type 4 connection . .	165
6.2.3 Configuring Content Manager Toolkit for DB2 type 2 connection . .	166
6.2.4 Using the JDBC Universal Driver for DB2 type 2 connection . . . .	166
6.2.5 Details on DB2 Universal JDBC Driver setup and tailoring. . . . .	169
6.3 System administration client installation . . . . .	169
6.3.1 Installation process . . . . .	170
6.4 Content Manager installation verification . . . . .	175
6.4.1 Troubleshooting . . . . .	175
<b>Chapter 7. Migrating from ImagePlus OS/390. . . . .</b>	<b>177</b>
7.1 Introduction . . . . .	178
7.1.1 Terminology . . . . .	178
7.1.2 Considerations and planning aids. . . . .	179
7.1.3 Prerequisites . . . . .	183
7.2 Migrating ImagePlus for OS/390 . . . . .	184
7.2.1 Assessing requirements and defining migration approach . . . . .	184
7.2.2 Planning and preparing for migration . . . . .	187
7.2.3 Installing Content Manager on z/OS . . . . .	197
7.2.4 Performing migration. . . . .	200
7.2.5 Performing post migration activities . . . . .	214
7.3 Customization alternatives . . . . .	215
7.3.1 Modifying the display name for attributes . . . . .	215
7.3.2 Modifying the display name for item types . . . . .	217
7.3.3 Modifying the attributes in an item type. . . . .	219
<b>Chapter 8. Migration from CM V2.3 to CM V8.3 . . . . .</b>	<b>223</b>
8.1 Introduction . . . . .	224
8.2 Pre-migration steps . . . . .	224
8.3 Preparing migration jobs . . . . .	226
8.4 Running the migration jobs . . . . .	229

8.5 Validate the environment . . . . .	230
8.6 Post product migration . . . . .	230
<b>Chapter 9. Migration from multiplatform to z/OS . . . . .</b>	<b>231</b>
9.1 Introduction . . . . .	232
9.2 Things to consider before migration . . . . .	232
9.3 Migration process . . . . .	233
9.4 Database structure . . . . .	234
9.4.1 Library Server database . . . . .	234
9.4.2 Resource Manager database and object storage . . . . .	234
9.5 Backup, backup, backup... . . . .	235
9.6 Update existing CM system on multiplatforms . . . . .	235
9.7 Review database naming conventions . . . . .	235
9.8 Install Content Manager V8.3 system on z/OS . . . . .	236
9.9 Copy CM V7.1 Library Server database to z/OS . . . . .	236
9.9.1 Use DB2LOOK to extract DB2 table definitions . . . . .	237
9.9.2 Create Content Manager V2.3 database on z/OS . . . . .	238
9.9.3 Catalog CM V7 z/OS database on workstation . . . . .	239
9.9.4 Create Content Manager V2.3 tables on z/OS . . . . .	240
9.9.5 Unload data from multiplatforms Library Server database . . . . .	240
9.9.6 Load data into z/OS database . . . . .	241
9.9.7 Status at this point . . . . .	242
9.9.8 Alternative ways of moving a DB2 database . . . . .	242
9.10 Run Content Manager V8.3 migration jobs . . . . .	243
9.10.1 ICMMMI70 . . . . .	243
9.10.2 ICMMMI71 . . . . .	245
9.10.3 ICMMMI72 . . . . .	246
9.10.4 ICMMMI73 . . . . .	248
9.10.5 ICMMMI74 . . . . .	249
9.10.6 ICMMMI75 . . . . .	250
9.10.7 ICMMMI76 . . . . .	250
9.10.8 ICMMMI77 . . . . .	251
9.10.9 ICMMMI7x and ICMMMI8x . . . . .	251
9.10.10 Status for Content Manager 8.3 Library Server migration . . . . .	254
9.11 Migrate Object Server to Resource Manager . . . . .	254
9.11.1 Tools to migrate Object Server database . . . . .	255
9.11.2 Export Object Server data . . . . .	255
9.11.3 Load Resource Manager tables . . . . .	256
9.11.4 Update RMSEVER and RMACCESS tables . . . . .	256
9.11.5 Migration checkpoint . . . . .	259
9.12 Resource Manager and objects migration . . . . .	260
9.12.1 Update V8.3 Library Server definitions . . . . .	260
9.12.2 Validate access to Resource Manager . . . . .	262

9.12.3	Define z/OS Resource Manager (RM) to multiplatforms RM . . . .	264
9.12.4	Define migration rules . . . . .	266
9.12.5	Status after updates . . . . .	272
9.12.6	Setup migrator task . . . . .	273
9.12.7	Start migration of objects . . . . .	273
<b>Part 3.</b>	<b>Logs and advanced configuration topics . . . . .</b>	<b>277</b>
<b>Chapter 10.</b>	<b>Traces and logs . . . . .</b>	<b>279</b>
10.1	Introduction . . . . .	280
10.2	Library Server traces and logs . . . . .	280
10.2.1	Enabling trace for Library Server . . . . .	280
10.2.2	DB2 joblogs . . . . .	283
10.3	Resource Manager traces and logs . . . . .	283
10.3.1	Resource Manager trace . . . . .	284
10.3.2	Resource Manager DB2 trace . . . . .	284
10.3.3	HTTP server trace . . . . .	285
10.4	Client's traces and logs . . . . .	285
10.4.1	Client for Windows . . . . .	285
10.4.2	System administration client . . . . .	287
10.4.3	UDB traces and logs . . . . .	290
10.4.4	Content Manager Toolkit for z/OS traces and logs . . . . .	294
<b>Chapter 11.</b>	<b>Multiple Content Manager instances in the same LPAR . . . . .</b>	<b>295</b>
11.1	Introduction . . . . .	296
11.2	Multiple Library Servers in the same DB2 subsystem . . . . .	296
11.3	Multiple Resource Managers in the same DB2 subsystem . . . . .	300
11.4	Configure Content Manager clients . . . . .	303
11.5	XML export . . . . .	306
<b>Chapter 12.</b>	<b>Considerations when using sysplex and multiple LPARs . . . . .</b>	<b>307</b>
12.1	Parallel Sysplex overview . . . . .	308
12.2	Multiple LPARs accessing shared data . . . . .	308
12.3	HTTP servers and task control blocks . . . . .	309
12.4	Resource Manager load on multiple HTTP servers . . . . .	310
12.5	Load balancing in a sysplex environment . . . . .	311
12.6	TCP/IP, Virtual IP Address, and workload balancing . . . . .	312
12.7	Object ownership in a sysplex environment . . . . .	312
12.8	Additional references . . . . .	313
<b>Chapter 13.</b>	<b>Access to DB2 plans and packages for Content Manager . . . . .</b>	<b>315</b>
13.1	Introduction . . . . .	316
13.1.1	Why you may want to alter the defaults . . . . .	316
13.1.2	Names used for DB2 plans . . . . .	316

13.2 Library Server packages and plans . . . . .	316
13.2.1 Library Server DB2 packages . . . . .	316
13.2.2 Library Server DB2 plans . . . . .	317
13.3 Grant access to Library Server DB2 components . . . . .	318
13.3.1 Library Server access . . . . .	320
13.3.2 CM clients logon processing to DB2 . . . . .	321
13.3.3 Batch jobs ICMMACL and ICMMDFUR . . . . .	322
13.3.4 Options used when binding Library Server programs . . . . .	323
13.3.5 Access to the DB2 views on Library Server tables . . . . .	324
13.4 Resource Manager packages and plans . . . . .	325
13.4.1 RM and LS on the same DB2 system . . . . .	326
13.4.2 RM and LS on separate DB2 systems . . . . .	328
13.5 Grant access to RM packages and plans . . . . .	332
13.5.1 Resource Manager batch programs . . . . .	332
13.5.2 Resource Manager CGI program running in the HTTP server . . . . .	333
13.5.3 Summary for RM GRANT specification . . . . .	334
13.6 LS problems related to access security . . . . .	335
13.6.1 Invalid values for connect user ICMCONCT . . . . .	335
13.7 RM problems related to access security . . . . .	337
13.7.1 Authority to write to DSNTRACE . . . . .	337
13.7.2 Authority to load and execute the CGI program in HTTP server . . . . .	338
13.7.3 Invalid RM user ID or password in LS configuration . . . . .	339
<b>Part 4. Appendixes . . . . .</b>	<b>341</b>
<b>Appendix A. WebSphere EAR file creation . . . . .</b>	<b>343</b>
Prerequisites to create the EAR file . . . . .	344
Purpose of the EAR file . . . . .	344
Create the EAR file . . . . .	345
Create a Web project . . . . .	345
Download and import the library files . . . . .	348
Add the HTML and Java files to your project . . . . .	351
Define your Web servlet in the project . . . . .	353
Step 9: Create an EAR file . . . . .	354
<b>Appendix B. WebSphere application setup . . . . .</b>	<b>355</b>
Prerequisites for WebSphere integration . . . . .	356
Purpose of WebSphere integration . . . . .	356
Components of WebSphere integration . . . . .	356
WebSphere configuration . . . . .	357
WAS enterprise server configuration . . . . .	358
WAS enterprise application configuration . . . . .	362
WAS HTTP server configuration . . . . .	369
Test the application . . . . .	371

WAS operational information . . . . .	372
<b>Appendix C. Sample code to change default collections . . . . .</b>	<b>375</b>
Source code for changing default collections . . . . .	376
<b>Appendix D. Migration error codes and messages . . . . .</b>	<b>379</b>
Migration error codes and messages . . . . .	380
<b>Appendix E. List and validate Content Manager definitions via SQL. . .</b>	<b>381</b>
Introductions . . . . .	382
List users and their default settings . . . . .	382
List item type definitions and DB2 table names . . . . .	383
List access modules if you use CM V8.1 or V8.2 . . . . .	386
Validate Library Server and DB2 table definitions . . . . .	387
Validate Library Server and DB2 view definitions . . . . .	390
<b>Related publications . . . . .</b>	<b>395</b>
IBM Redbooks . . . . .	395
Other publications . . . . .	395
Online resources . . . . .	397
How to get IBM Redbooks . . . . .	398
Help from IBM . . . . .	398
<b>Index . . . . .</b>	<b>399</b>



# Figures

1-1	Simple Content Manager for z/OS system architecture . . . . .	6
2-1	Content Manager for z/OS system architecture . . . . .	15
2-2	Create auto-linking . . . . .	25
2-3	Adding document part in item type definition . . . . .	26
2-4	Define versioning . . . . .	28
2-5	Creating item type subsets . . . . .	30
2-6	Create database index . . . . .	32
3-1	Output of the sample HTML file . . . . .	70
4-1	Definition of document routing user exit in a work basket . . . . .	94
4-2	Definition of document routing user exit in a collection point . . . . .	95
4-3	Definition of a document routing user exit in a business application . . . . .	96
5-1	System administration client: User details for maximum privilege set . . . . .	117
5-2	System administration client: Defaults for RM-related options . . . . .	118
5-3	Enable ACL user exit program from the system administration client . . . . .	127
5-4	Processing store request and override default collection as required . . . . .	148
5-5	Returning updated collection information to Library Server . . . . .	149
6-1	Creating an application environment . . . . .	155
6-2	Install definition . . . . .	156
6-3	Content Manager installation: Custom install System Adm Client . . . . .	170
6-4	System administration client connection setup . . . . .	171
6-5	Configuration screen shot . . . . .	173
6-6	Log file containing the output of running cmcfigls -t comtypes . . . . .	174
6-7	Log file containing the output of running cmcfigls -t predefs . . . . .	175
6-8	Error message encountered . . . . .	176
7-1	Results from ICMSTMAXKEYWORD DB2 query . . . . .	192
7-2	Default attribute properties in system administration client . . . . .	216
7-3	Attribute after change in system administration client . . . . .	217
7-4	IPFOLDER highlighted in system administration client . . . . .	218
7-5	Selecting properties for IPFolder item type . . . . .	218
7-6	Default IPFOLDER display name . . . . .	219
7-7	Customized folder display name . . . . .	219
9-1	User defaults migrated from Library Server on multiplatforms . . . . .	260
9-2	Migrated item type definition initially points to RM on multiplatforms . . . . .	261
9-3	Test connection to Resource Manager on z/OS . . . . .	262
9-4	Failed request to Resource Manager server on z/OS . . . . .	263
9-5	Successful reply from a Resource Manager server on multiplatforms . . . . .	263
9-6	View the source of the reply from Resource Manager . . . . .	264
9-7	List of servers initially known to RM on multiplatforms, RMPALM . . . . .	265

9-8	Define z/OS RM to the RM on multiplatforms . . . . .	266
9-9	Work with storage classes for RM on multiplatforms . . . . .	267
9-10	Enter details for the new storage class on remote destination on z/OS268	
9-11	New ITSO8ZOS storage class has been created . . . . .	269
9-12	Select the management class and show its properties . . . . .	270
9-13	Update the migration policy . . . . .	270
9-14	Change the retention period from Forever to something less . . . . .	271
9-15	Updated migration policy information. . . . .	271
9-16	Select the target remote storage class . . . . .	272
10-1	Library Server Configuration: Log and trace tab . . . . .	281
10-2	Log file output . . . . .	283
10-3	Client for Windows log preference setting . . . . .	286
10-4	System administration client: Log configuration utility . . . . .	287
10-5	System administration client: System administration event logging . . . . .	289
10-6	System administration client: Enabling item events logging . . . . .	290
11-1	Server configuration utility . . . . .	304
12-1	Multiple LPARs accessing shared database . . . . .	309
A-1	Create a Web project. . . . .	346
A-2	Provide a project name . . . . .	346
A-3	Specify J2EE settings . . . . .	347
A-4	Import external libraries . . . . .	349
A-5	Add HTML file to the project . . . . .	351
A-6	Add Java source code in the Web project . . . . .	352
A-7	Define Web servlet in the project. . . . .	353
A-8	Create EAR file . . . . .	354
B-1	WebSphere Application Server administrative console login. . . . .	357
B-2	WAS application server setup . . . . .	358
B-3	New application server setup. . . . .	359
B-4	Confirm new application server setup . . . . .	360
B-5	Java virtual machine setup . . . . .	361
B-6	New application installation preparation . . . . .	362
B-7	New application installation preparation: Bind app to virtual host . . . . .	363
B-8	New application installation setup . . . . .	364
B-9	New application installation setup: Map Web modules to virtual host . . . . .	365
B-10	New application installation setup: Map application to server . . . . .	366
B-11	New application installation: Summary . . . . .	367
B-12	New application installation: Save to master configuration . . . . .	368
B-13	New application installation: Save . . . . .	369
B-14	WAS HTTP server configuration: Host alias . . . . .	370
B-15	WAS HTTP server configuration: Update Web server plugin . . . . .	371



# Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:  
*IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.*

*The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:* INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

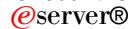

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

## COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

# Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

 eServer®	DB2®	MVS/ESA™
Redbooks (logo)  ™	DRDA®	Notes®
z/OS®	ImagePlus®	OS/390®
zSeries®	IBM®	Parallel Sysplex®
AIX®	IMS™	Redbooks™
CICS®	Language Environment®	RACF®
Distributed Relational Database Architecture™	Lotus Notes®	Tivoli®
Domino®	Lotus®	VisuallInfo™
DB2 Connect™	MO:DCA™	WebSphere®
DB2 Universal Database™	MQSeries®	
	MVS™	

The following terms are trademarks of other companies:

Java, JDBC, JDK, JSP, J2EE, Solaris, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft Internet Explorer, Microsoft, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, and service names may be trademarks or service marks of others.

# Preface

This IBM® Redbook covers IBM DB2® Content Manager Enterprise Edition for z/OS® Version 8.3 implementation, installation, and migration. It is intended to be used in conjunction with the product publications. In many areas, it focuses on the important topics and issues related to the planning and execution of these tasks rather than providing comprehensive information to perform the tasks. This IBM Redbook is aimed at architects, designers, developers, and system administrators of Content Manager systems.

In Part 1, we introduce Content Manager for z/OS Version 8.3 and provide information on how to implement a Content Manager for z/OS system. We cover general implementation topics and provide special z/OS-related topics including Content Manager Toolkit for z/OS (also referred to as host APIs), host integration, Content Manager security, and Content Manager exits.

In Part 2, we describe Content Manager for z/OS Version 8.3 installation. In addition, we cover migration of ImagePlus® OS/390®, Content Manager Version 2.3 for OS/390 (also known as Content Manager Version 7.1 for OS/390), and Content Manager Version 7 for Multiplatforms, to Content Manager for z/OS Version 8.3.

In Part 3, we cover how to activate traces and set up logs to maintain and troubleshoot your Content Manager for z/OS system. We discuss some advanced configuration topics, including configuring multiple Content Manager instances in the same logical partition (LPAR), issues and considerations when setting up a Content Manager system in the sysplex environment, and changing default access to DB2 plans and packages. At the end of the last chapter, we also look at error situations you may encounter related to security and where you can find information to help you analyze the situation.

By using this IBM Redbook in conjunction with the product publications, we hope you learn the basics that you will need to implement, install, or migrate a Content Manager system on the z/OS platform. We also hope you enjoy some of the hints and tips we provide throughout this redbook, including the information in the appendixes.

## The team that wrote this IBM Redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization, San Jose Center.

**Wei-Dong Zhu** (Jackie) is a Content Management Project Leader with the International Technical Support Organization at the Almaden Research Center in San Jose, California. She has more than ten years of software development experience in accounting, image workflow processing, and digital media distribution. She holds a Masters of Science degree in Computer Science from the University of Southern California. Jackie joined IBM in 1996. She is a certified Solution Designer for IBM DB2 Content Manager.

**Claus Heisterberg-Andersen** is a Certified Senior IT Specialist from IBM Denmark. He has more than 20 years of experience with IBM, with the last nine years in the Content Management field on both Multiplatform and z/OS. During this period, he has designed and developed solutions with customers integrating Content Manager with their line of business applications. He has written extensively on previous IBM Redbooks™ on Content Manager migration issues.

**Marcelo Takashi Souza** is an IT Specialist in Brazil. He has two years of experience in the Content Management Multiplatform and z/OS fields and five years experience in DB2 z/OS. His areas of expertise include IBM DB2 Content Manager, On Demand, and DB2 z/OS. He has worked at IBM for five years in the IBM Support Center (post sales) and participated in implementation services for the above software.

**Jaco van Straaten** is an IT Technical Specialist in Cape Town, South Africa, at the life insurance company Sanlam. He has ten years of experience in the document management field. He holds a B.Sc. degree in Computer Science and applied mathematics from the University of Stellenbosch. His areas of expertise include IBM DB2 Content Manager, Information Integrator for Content, and ImagePlus. He has experience in both z/OS and the Multiplatform environments. He also has extensive development experience using Java/J2EE™ and C/C++.

**Rob Weaver** is a Senior Software Engineer, working for the Content Manager development lab leading the Content Manager z/OS system test (SVT) team. Rob has almost 28 years with IBM, and has spent the last 13 years in various jobs developing and supporting ImagePlus and Content Manager. His skills with Content Manager span multiple platforms, but his primary area of expertise is z/OS (mainframe). He has participated in previous redbook projects that dealt with ImagePlus implementation, and currently works closely with account teams and their customers in developing strategies for migrating from ImagePlus to Content Manager.

Thanks to the following people for their contributions to this project:

Emma Jacobs  
IBM International Technical Support Organization, San Jose Center

Rong-hoang (Rick) Chang

Theresa Dain  
Glen Dinsmore  
Ed Gallagher  
Esther Hollinger  
Nguyen Phan  
Dwayne Richardson  
Larry Schroeder  
Chun-Fu Su  
Ali Wasti  
Peter Wansch  
IBM Software Group/Information Management Solutions, US

Gerhard Fichtinger  
IBM Global Services/Application Management Services, Germany

Michael Heyl  
IBM Sales & Distribution/Product Introduction and Exploration, Americas

Jørn Borup-Andersen  
IBM Sales & Distribution/Software Sales, Denmark

## Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll team with IBM technical professionals, Business Partners, and/or customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

<http://www.ibm.com/redbooks/residencies.html>

## Comments welcome

Your comments are important to us!

We want our IBM Redbooks to be as helpful as possible. Send us your comments about this or other Redbooks in one of the following ways:

- ▶ Use the online **Contact us** review redbook form found at:  
<http://www.ibm.com/redbooks>
- ▶ Send your comments in an e-mail to:  
[redbook@us.ibm.com](mailto:redbook@us.ibm.com)
- ▶ Mail your comments to:  
IBM Corporation, International Technical Support Organization  
Dept. QXXE Building 80-E2  
650 Harry Road  
San Jose, California 95120-6099



# Part 1

# Introduction and implementation

In this part, we introduce IBM DB2 Content Manager for z/OS V8.3 and provide information on how to implement a Content Manager for z/OS system. We discuss Content Manager Toolkit for z/OS (also referred to as host APIs), host integration, Content Manager security, and Content Manager exits.

Archived



# Introduction

In this chapter, we introduce IBM DB2 Content Manager for z/OS.

We cover the following topics:

- ▶ Background
- ▶ What does Content Manager do?
- ▶ Content Manager system architecture overview
- ▶ Enhancements in Content Manager for z/OS V8.3
- ▶ Differences between ImagePlus OS/390 and CM V8.3
- ▶ Differences between CM for z/OS and Multiplatforms

In Chapter 2, “Implementation” on page 13, we provide more information about the product to help you plan and implement a Content Manager solution.

## 1.1 Background

Over the last decade, there has been widespread adoption of e-business solutions. Companies need to manage information, processes, knowledge, and business operations electronically; the systems that manage these areas have been developed over time.

These systems have been known variously as content management, document management, knowledge management, collaboration management, digital asset management, and digital rights management. IBM has been a leader in providing solutions for several of these areas.

IBM DB2 Content Manager is one of the key products IBM has developed to meet the needs of industry. In this chapter, we provide a brief introduction to the product. In the remaining chapters of this part, we describe the product in more detail and discuss how you can implement a Content Manager solution.

## 1.2 What does Content Manager do?

IBM DB2 Content Manager for z/OS (Content Manager) is a product designed to manage volumes of digital content and support mission critical business processes. Content Manager is a production strength content management system designed to capture, process, and store a vast amount of converted paper-based information. It is also used for rich media management.

Content Manager is a highly scalable repository for virtually any type of digital content, including HTML- and XML-based Web content, document images, electronic office documents, and rich media such as digital video and audio. A single Content Manager system can support multiple content stores distributed across the enterprise or across the Internet. This allows content to be stored close to its point of use while remaining under central management control, reducing bandwidth requirements, and increasing disaster protection.

Unlike simple file systems, Content Manager uses a powerful relational database to provide indexed search, security, and granular access control at the individual content item level. Content Manager provides check-in and check-out capabilities, version control, object-level access control, a flexible data model that enables compound document management, and advanced searching based on user-defined attributes. It also includes workflow functionality, automatic routing, and tracking content through a business process according to predefined rules.

## 1.3 Content Manager system architecture overview

A Content Manager solution is always composed of *DB2 database*, one Library Server, one to many Resource Managers, and multiple clients.

*Library Server* contains and manages the system configuration and the metadata of all the content. It is built on *DB2 stored procedures* and has its own database.

*Resource Manager* manages content object storage. It interfaces with clients with *HTTP server*. Storage management systems of Resource Manager can be either Object Access Method (OAM) or Tivoli® Storage Manager (TSM).

Multiple clients are used to support system administration via *system administration client* and day-to-day operations via *Client for Windows®*.

When content is created and stored, it is physically stored on a Resource Manager. Content metadata and access control is managed by Library Server. Clients store or retrieve content via Library Server. When a client requests to retrieve an object, Library Server performs the query against its database, and passes the result back to the client with the object token and resource location for which the user is authorized. The client then communicates directly with Resource Manager to retrieve the object, using standard protocols.

Figure 1-1 shows a simple Content Manager for z/OS system architecture.

In Chapter 2, “Implementation” on page 13, we cover each component in more detail. In addition, we address important concepts such as Content Manager data modeling, user access definition, workflow, and object storage.

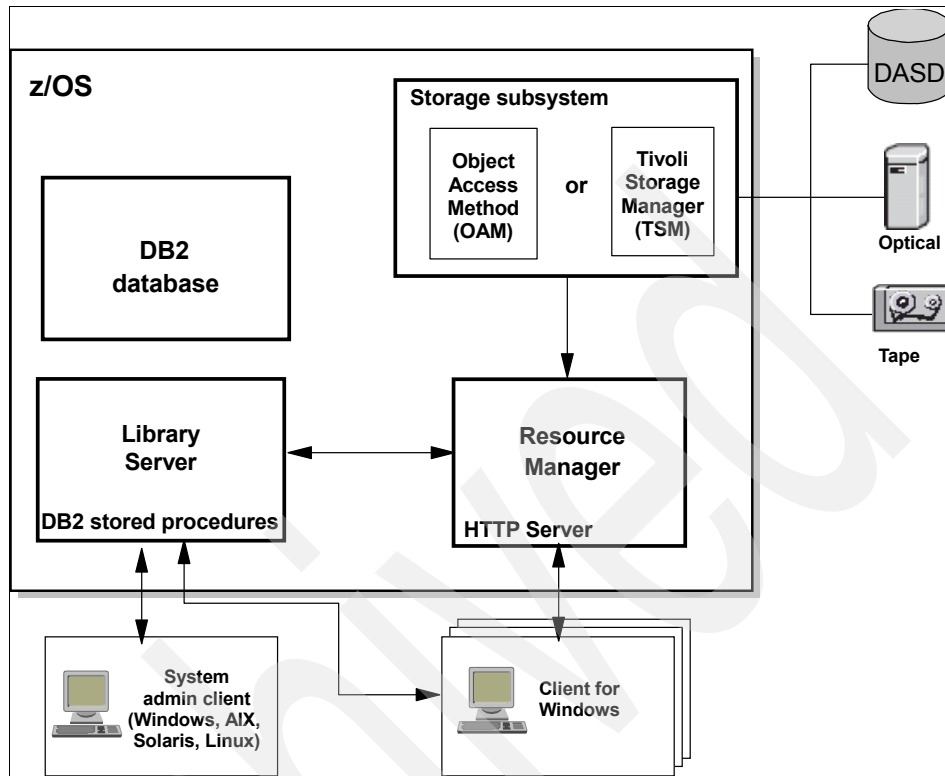


Figure 1-1 Simple Content Manager for z/OS system architecture

## 1.4 Enhancements in Content Manager for z/OS V8.3

The enhancements added from Content Manager for z/OS V2.3 (also known as V7.1) to Content Manager for z/OS V8.3 are:

- ▶ Support for TSM application programming interfaces
- ▶ Content Manager Toolkit for z/OS
- ▶ Improved performance
- ▶ Common system administration and Windows client
- ▶ z/OS UNIX System Services support (USS)
- ▶ XML support
- ▶ Document routing enhancements
- ▶ Common architecture for logging and tracing

## **Support for TSM application programming interfaces**

Content Manager for z/OS V8.3 added support for Tivoli Storage Manager (TSM) in z/OS. Now Resource Manager for z/OS can store either OAM or TSM. Storing objects to Resource Manager on z/OS and a TSM collection increases the maximum object size currently supported.

## **Content Manager Toolkit for z/OS**

Content Manager Toolkit for z/OS provides Java™ connector support z/OS UNIX® System Services (USS).

The toolkit provides connectivity to Content Manager systems running on all supported platforms. Clients include Java applications that run in batch under z/OS using Enterprise COBOL, UNIX, and Web applications that run in the WebSphere® for z/OS application server.

## **Improved performance**

Library Server takes advantage of DB2 stored procedures, which reduce network traffic. By taking advantage of DB2 stored procedures and less network traffic, Library Server is able to improve performance and scalability. Resource Manager for z/OS is implemented as a Common Gateway Interface (CGI) program running on the IBM HTTP Server. WebSphere Application Server for z/OS is not required.

## **Common system administration and Windows client**

The system administration client provides access to Content Manager for z/OS or Multiplatform configuration and setup. Windows client provides an out-of-the-box, user interface to access and manage the Content Manager system.

## **z/OS UNIX System Services support (USS)**

Content Manager for OS/390 V2.3 requires CICS®. With Content Manager for z/OS V8.3, Resource Manager and toolkit utilize USS; CICS is not required.

## **XML support**

Content Manager V8.3 system administration client enables you to export your system administration data into an XML readable file and import data from an XML file into the system. This capability allows you to copy administrative settings from one server to another by exporting the information and importing it into the system. For example, you can easily port a development Content Manager system setup to a testing environment.

## Document routing enhancements

Content Manager document routing is enhanced in V8.3 to include decision points, actions, action lists, parallel routing, and user exit support. In addition, a new graphical builder within the system administration client helps you easily define your document routing processes.

## Common architecture for logging and tracing

Content Manager V8.3 provides a common log and trace architecture that covers most system components:

- ▶ The system administration client now provides the log control utility, which you can use to set log and trace parameters for multiple system components.
- ▶ A single directory for all component log files.
- ▶ A standard log file timestamp format using Greenwich Mean Time.
- ▶ Logging information can be set to a single user ID.
- ▶ A unique log ID that is common across different system component log files.

# 1.5 Differences between ImagePlus OS/390 and CM V8.3

At a very high level, with a narrow view, ImagePlus and Content Manager perform similar tasks. Both solutions facilitate the management of electronic content. At a lower level, with a wider view, the two solutions are, in reality, radically different.

In this section, we cover the major differences as follows:

- ▶ Differences in content support
- ▶ CM Resource Manager versus ImagePlus IODM
- ▶ CM Library Server versus ImagePlus IPFAF
- ▶ Differences in the setup of folders, documents, and attributes

## Differences in content support

ImagePlus generally supports the management of stored documents, MODCA or TIFF, and other electronic data commonly referred to as *coded data*, or IOCA, which is a subset of MODCA. The processes for ingesting data into ImagePlus inherently limit the variety of data that can be stored. Content Manager, on the other hand, provides the capability, out-of-the-box, to store virtually any electronic data you may have.

The differences between ImagePlus and Content Manager go much deeper than just the data that each manages.

## ImagePlus system background

ImagePlus is a z/OS only solution and utilizes the OAM component of z/OS to facilitate the storage management of the actual object content. Note that Content Manager on z/OS also utilizes OAM. The ImagePlus solution is made up of two primary components. The base component is the *ImagePlus Object Distribution Manager*, IODM. The second component, *ImagePlus Folder Application Facility*, IPFAF, is available as two separate functions. The base function for IPFAF is the Application Programming Interface, IPFAF/API, which provides the application interface layer. IBM provides another layer on top of IPFAF/API, the Folder Workflow Application, IPFAF/FWA, which provides the 3270 green-screen interface that you see.

IODM manages the communications among the IPFAF/API layer, OAM, and the ImagePlus Workstation Program, IWPM, used for rendering objects. IPFAF can run in either an IMS™ or CICS environment. IODM is supported under CICS only.

## CM Resource Manager versus ImagePlus IODM

Content Manager consists primarily of two components, Resource Manager and Library Server.

The base component, as compared to ImagePlus, is *Resource Manager*, which also interfaces with OAM in a z/OS environment. Unlike IODM, Resource Manager is not limited to z/OS, but can run on Windows, SUN, AIX®, or Linux®. On the non-z/OS platforms, the operating system file services provide some of the content management support. TSM can also be implemented to manage retention and migration to and from DASD.

## CM Library Server versus ImagePlus IPFAF

The corresponding IPFAF component in Content Manager is *Library Server*. Library Server, as with Resource Manager, is not limited to the z/OS platform, but is supported on all the same platforms as Resource Manager, and does not need to be on the same platform as Resource Manager. Library Server is essentially a DB2 database which is managed by a series of DB2 stored procedures called by an application programming interface (API) layer from a system administration client, Windows client, or an eClient enabled via a mid-tier server. Additionally, unlike IPFAF, Library Server does not require IMS or CICS; it just requires DB2.

ImagePlus supports casual scanning with the IWPM client, along with high volume input from a variety of sources. The input is indexed and stored using either the batch store process or online auto-indexing, also known as Scan Notification Processing (SNPA). Content Manager Windows client provides casual scanning support; but unlike ImagePlus, it also supports storing objects

other than MODCA or TIFF, such as Word documents, PDF files, JPEG images, or virtually any format available.

Content Manager does not directly provide any high volume input capabilities such as ImagePlus provides. Many Business Partner solutions provide high volume or auto-index processing. Several Business Partners have enhanced their existing offerings, which support ImagePlus, to also support Content Manager, therefore, making a transition to Content Manager smoother and more consistent with existing approaches.

### **Differences in the setup of folders, documents, and attributes**

The differences between ImagePlus and Content Manager discussed so far are very significant, but the differences go deeper yet. ImagePlus has a set group of attributes with a rigid data model of folders and documents, requiring that every document is in a folder. Content Manager provides much greater flexibility in this area. With Content Manager, a system administrator defines the attributes, along with their characteristics, that make sense to the customer's needs. These characteristics include all of the relative properties of an attribute such as name, data type, and data range. The administrator can then create Content Manager item types to categorize, or group data. For example, an item type for documents can be defined with exactly the attributes that make sense for a specific environment. Unlike ImagePlus, this document item type can stand alone and provide all the input, search, and manipulation capabilities you may need. If grouping documents in a folder is the best solution, a folder item type with only the necessary attributes can be defined and have documents automatically linked to them. If grouping folders into higher level folders is necessary, this can also be accomplished with Content Manager.

## **1.6 Differences between CM for z/OS and Multiplatforms**

The main differences between Content Manager for z/OS and Content Manager for Multiplatforms are that certain functions are not yet available in Content Manager for z/OS.

### **Differences in Library Server**

The following features of Library Server are not available in z/OS:

- ▶ Text searching on document objects
- ▶ Oracle support

Note, Library Server uses DB2 database functionality to accomplish text searching on multiplatforms. The text searching is available in DB2 for z/OS V8,



not V7. Library Server in the z/OS environment has not taken advantage of the text search capability in DB2 for z/OS V8.

### **Differences in Resource Manager**

The LAN caching feature of Resource Manager is not available in z/OS.

Note, Resource Manager for Multiplatforms uses IBM WebSphere Application Server. Resource Manager for z/OS, however, is implemented as a CGI program running on the IBM HTTP server.

### **Functionality not available in Content Manager Toolkit for z/OS**

The following features of Content Manager Toolkit for z/OS are not yet available (but they are available for Content Manager Toolkit for Multiplatforms):

- ▶ Java visual beans
- ▶ C/C++ support in the z/OS environment
- ▶ Web services support

### **Information Integrator for Content**

Information Integrator for Content is not available on z/OS. You need to install Information Integrator for Content on a Multiplatform system and then point it to Content Manager for z/OS in order to use it.



# Implementation

In this chapter, we describe the Content Manager for z/OS implementation process from a planning and design point of view.

We cover the following topics:

- ▶ Analyze business requirements
- ▶ Plan system topology
- ▶ Define data model
- ▶ Implement user access
- ▶ Implement workflow
- ▶ Manage object storage
- ▶ Manage different domains
- ▶ Provide Web access
- ▶ Backup and recovery
- ▶ Performance tuning
- ▶ Additional products that use Content Manager

We focus on the concepts behind the implementation rather than the detailed instructions to perform the actual implementation. For example, we discuss when and why to implement some functionality in Content Manager for z/OS and what to consider before you attempt to do it.

When appropriate, we refer you to other publications that describe the detailed instructions for each of the topics.

## 2.1 Analyze business requirements

Your first step should always be to analyze and document your business requirements. It is very important for planning the installation of Content Manager for z/OS or the migration to it. Analyzing business requirements and planning ahead help you create a system that meets and grows with your business requirements.

The following are key questions you should consider during the analysis phase:

- ▶ What type of content or documents need to be stored in Content Manager?
- ▶ What is the size of each of the document types?
- ▶ How many documents will be stored per month?
- ▶ Will the rate that documents are stored per month increase, and if so, at what rate?
- ▶ How long do you need to store these documents? Are there any legal requirements?
- ▶ How can you logically organize or group the documents in your organization?
- ▶ Are there multiple departments in your organization that need to access Content Manager?
- ▶ Which users are allowed to store, update, or delete items stored in Content Manager?
- ▶ Do you need to save older versions of documents?
- ▶ What are your workflow requirements?
- ▶ Do you need any data migration?
- ▶ How many users will access the system?
- ▶ Do your users require Web access?
- ▶ Is there any integration required to external applications?
- ▶ What type of security is required?
- ▶ What is your existing infrastructure? For example, do you use RACF®?

Refer to the following IBM Redbook for more information regarding this topic:

- ▶ *Content Manager Implementation and Migration Cookbook*, SG24-7051

The above redbook covers Content Manager V8.2, but it is applicable to both Multiplatforms and z/OS for V8.3.

## 2.2 Plan system topology

You need to understand each component of DB2 Content Manager for z/OS before you can design your system topology. The following is a list of the components:

- ▶ DB2 database
- ▶ Library Server
- ▶ Resource Manager
- ▶ Storage subsystem
- ▶ HTTP server
- ▶ System administration client
- ▶ Windows client (Client for Windows)
- ▶ Web interface
- ▶ Content Manager Toolkit for z/OS

Figure 2-1 provides Content Manager for z/OS system architecture and shows how most of these components fit in the entire system topology.

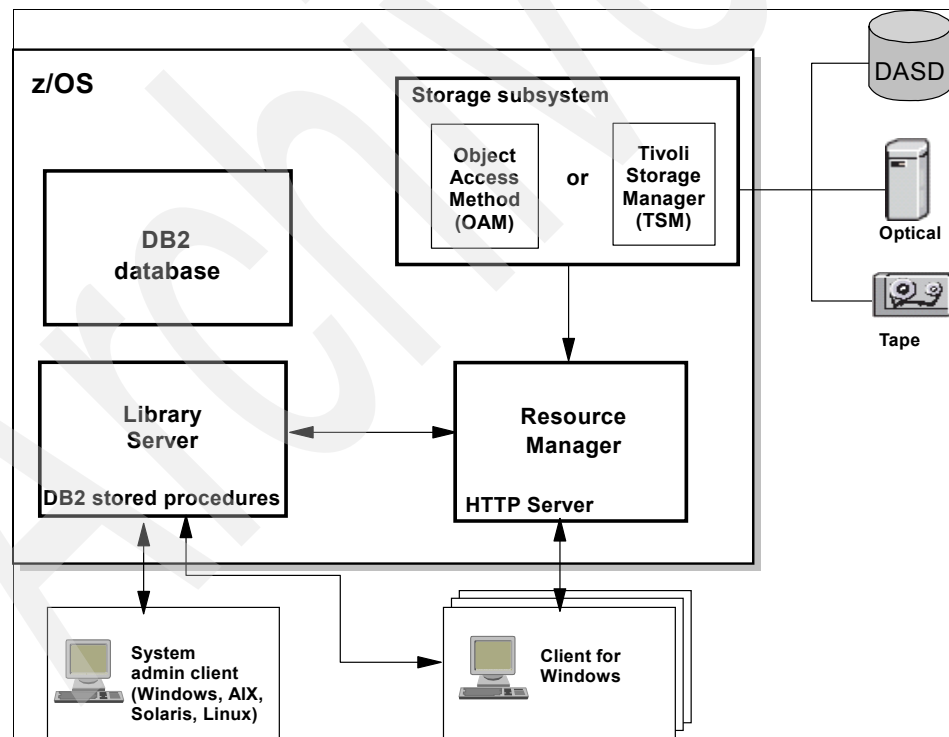


Figure 2-1 Content Manager for z/OS system architecture

## 2.2.1 DB2 database

IBM DB2 is a relational database management system. It is used to store, search, and retrieve content or documents within the Content Manager system.

A Content Manager system requires two databases, one for the Library Server (default is LSDB) and one for the Resource Manager (default is RMDB), whether Library Server and Resource Manager run on the same machine or on separate machines.

If the storage subsystem is OAM, it requires an additional database.

Content Manager can also be configured to use Oracle as a database; but this configuration is not possible on z/OS.

Also note is that the DB2 run-time client does not have the capability to connect to a DB2 database that runs on z/OS. In this case, you have to configure a DB2 Connect™ as a gateway for the DB2 run-time client. Another option is to use DB2 Connect Personal Edition to directly access a DB2 z/OS database.

Both Resource Manager and Content Manager Toolkit for z/OS need connectivity to the database from the z/OS UNIX System Services (USS) side.

## 2.2.2 Library Server

The primary purpose of Library Server is to service client requests for content. It is used to define information or content that you want to store.

Library Server manages the access control to the content or documents stored in Resource Manager.

There can only be one Library Server in each Content Manager system (instance). If you have more than one Library Server, then they do not connect in any way and they run as individual systems.

You may install more than one Library Server on a single z/OS system. This is normally done when you want to use a test system.

Library Server may be linked to more than one Resource Manager. A second Resource Manager can be on another machine and even on another platform, for example, AIX.

## 2.2.3 Resource Manager

The primary purpose of Resource Manager is to store and retrieve content objects or documents.

The physical objects stored in Resource Manager for z/OS are managed by the storage subsystem that can be either OAM or TSM in z/OS. Resource Manager stored objects may reside in a DB2 database on DASD, tape, or any media supported by OAM or TSM.

The HTTP server is used as an interface to service requests from clients to Resource Manager. Resource Manager for Multiplatforms needs WebSphere Application Server as its interface. Resource Manager for z/OS needs only HTTP Server.

In the most basic configuration, you have one Library Server and one Resource Manager. Both connect to the same DB2 subsystem on the same z/OS mainframe.

You may extend this configuration by adding an additional Resource Manager that either runs on z/OS or on Multiplatforms. All Resource Managers link to the single Library Server.

This is normally done if you have users in more than one city, state, or country, and you want to store or serve documents in or from a repository closer to users. This is done to optimize network access.

You can replicate the information and data stored in one Resource Manager to another Resource Manager. You can also use a second Resource Manager to split the load or overhead in a very large organization.

## **2.2.4 Storage subsystem**

The storage subsystem is used to manage the physical documents stored in Content Manager.

There are two storage management systems available: Object Access Method (OAM) or Tivoli Storage Manager (TSM). You can use either OAM, TSM, or both for your Resource Manager.

## **2.2.5 HTTP server**

The purpose of the HTTP server is to act as an interface between the clients and Resource Manager. Documents are stored and retrieved from Resource Manager using HTTP and HTTPs.

In a normal configuration, each Resource Manager has its own HTTP server. Resource Manager for Multiplatforms also requires a WebSphere Application Server that runs on top of the HTTP server. Resource Manager for z/OS does not require WebSphere Application Server.

## 2.2.6 System administration client

The system administration client is used to define the configuration of Library Server and Resource Manager. It is also used to define the document model and the workflow. In addition, you define users and manage access control using the system administration client.

You need to install the system administration client on a Windows workstation even if you use Content Manager for z/OS.

You can install more than one system administration client in your organization. This is normally done when you define administration domains. You use these when you have multiple departments that want to administer their own users, data model, or workflow.

## 2.2.7 Windows client (Client for Windows)

The Windows client of Content Manager is an out-of-the-box application that can be used to store or retrieve content or documents from Content Manager for z/OS. It also has an advanced search option and an interface to manage workflow or document routing.

You can configure the Windows client to connect to more than one Library Server; however, normally, you may connect to only one Library Server at the same time.

If you have a Content Manager Information Integrator for Content federated server installed, using the federated connection, it is possible for you to connect to multiple Content Manager V8 Library Servers and even IP/390 servers.

You can develop your own custom Windows client by using the client APIs or you can extend the Content Manager Windows client with the client user exits.

You do not need to configure a connection between Resource Manager and the Windows client.

## 2.2.8 Web interface

No Web interface is provided with Content Manager for z/OS.

If you require Web access, you may use IBM DB2 Content Manager Information Integrator for Content with eClient. This is only available on Multiplatforms. In this case, you need to configure a connection to Content Manager for z/OS and then let the eClient connect to it. The eClient requires WebSphere Application Server because it is a WebSphere application.



You can develop your own Web applications by using WebSphere Application Server and your own Java code that calls the Content Manager Toolkit for z/OS.

If you have more than one Library Server, your Web application has to make a connection to each Library Server. This is also true for eClient.

The Web application or eClient is configured to link to Library Server using SQL calls and to Resource Managers with HTTP requests.

## 2.2.9 Content Manager Toolkit for z/OS

Content Manager Toolkit for z/OS (also referred to as host APIs) is provided on the z/OS UNIX System Services side of z/OS. The APIs connect to Library Server which is responsible to make the requests to Resource Manager.

You can use Content Manager Toolkit for z/OS to integrate your business applications with Content Manager for z/OS. Note, you can also use the user exits of Content Manager to integrate with your business applications.

The following publication provides more information on the components of Content Manager for both Multiplatforms and z/OS:

- ▶ *IBM DB2 Content Manager Enterprise Edition V8.3: Planning and Installing Your Content Management System*, GC27-1332

The next publication provides more information from a z/OS perspective:

- ▶ *IBM DB2 Content Manager for z/OS: Planning and Installing Your Content Management System*, GC18-7698

Once you understand different components of Content Manager for z/OS and your business requirements, it should be easy to design Content Manager topology suitable for your organization.

## 2.3 Define data model

The data model in Content Manager is used to define the structure and relationships of objects stored in it. You need to plan and design your data model before you can implement Content Manager. You also need to understand the building blocks or data model entities that can be used in Content Manager to define your data model.

### 2.3.1 Plan your data model

This is one of the most important planning activities you have to do to implement a Content Manager system. The definition of the data model has an impact on almost every aspect of Content Manager. Before starting the data modeling, we must first understand the concepts and objectives.

All objects stored in Content Manager are organized or grouped together using the data model. The data model also defines the user interaction with the objects when a user searches for an item or navigates through the hierarchical levels of items.

The relationships between the item types and the hierarchical structure in the data model are reflected in the interface of both the Windows client and the eClient.

The data model is also used to define the access control of the user to the items or objects stored in Content Manager.

We recommend using the following steps when planning your data model:

1. Identify the content or data that you want to store.

The following are some examples of content:

- Scanned application forms
- Electronic documents
- Digital photos
- E-mails sent from clients
- Audio conversations

2. Separate the data into operational and non-operational data.

*Operational documents* are used to perform business processes. An example of this type of document is a scanned policy application form.

Operational documents can also be used in workflow processes and they normally require actions from a user, for example, a change of address request document that requires address changes for a policy holder.

*Non-operational documents* are used to provide information about business processes. For example, a document that describes the rules of an insurance policy is a non-operational document.

3. Sort or group together the data that is similar into *item types*.

The purpose of this step is to get a list of item types that you can define in your data model. You can sort your content data by using the following criteria:

- The type of content
- The purpose of the content
- The type of customer to which it relates
- The users that access it
- The department in your organization to which it belongs
- Data that has repeating groups

Grouping or identifying the item types can also be explained with the following example of an insurance company:

- a. In an insurance company, you start with a customer. All customer-related documents can be grouped together. For example, you can create a customer item type. You can then group the copy of the customer's passport or the health history of the customer to the customer item type.
  - b. The next item type is related to the insurance products of a customer. For example, a customer might have a vehicle insurance policy and a life insurance policy. The documents related to each of these products can be stored using different item types.
  - c. The documents related to an action of a customer on a product can also be stored using a different item type. For example, a customer wants to make a claim on a vehicle insurance policy. All documents related to claims can be a claim item type.
4. List the attributes of each item type.

An *attribute* is used to store a value that describes a property of an item type. For example, the first name and last name can be two attributes for a customer (item type).

The value of an attribute can later be used to locate the stored object (or items) when a search is performed by a user.

Attributes can be grouped together by using *attribute groups*. This can be used for the address of a customer that may contain attributes for the street, city, and country.

5. Identify the hierarchical relationships between item types.

In this step, you have to define the relationships between your item types. The relationships are normally defined in a hierarchical structure.

The following *link types* are provided in Content Manager:

- A reference attribute
- A foreign key
- A link

A *reference attribute* is a single-direction association between a root or child item type and another root item type. For example, in the insurance company, you have a vehicle item type and an underwriter item type. You can use

reference attribute to link the vehicle to the underwriter. Note, we discuss root and child item types (or components) later in 2.3.2, “Data model entities” on page 22.

In DB2 database, you use *foreign keys* to enforce referential integrity among tables. In Content Manager system, foreign keys are used to link item types with external tables. For example, in the vehicle insurance item type, you have an attribute that identifies the agent that sold the policy to the client. This attribute is the agent’s employee number. It is a foreign key to an external DB2 employee table.

A *link* is used to associate two item types with each other. In Content Manager, links between item types are implemented by using *auto-linking*. Links can be used to associate child and root item types. Auto-linking is discussed in more detail in “Auto-linking” on page 24.

6. Identify users that have access to each item type.

The next step is to identify the users that should have access to each item type. This information can be used to refine your data model.

You also need to document what type of access users require for each item type and always work with user groups. For example, the claims department needs access to create a claim item type, but it does not require access to create a customer item type.

7. Identify the searchable attributes of each item type.

To further refine your data model, the last step is to identify which attributes are frequently used for searching. You can define the commonly used searchable attributes as indexes to improve your system performance.

## 2.3.2 Data model entities

Now that we have discussed how to plan for your data model, let us review various data model entities that make up the data model of a Content Manager system. Some of them repeat what you have learned from the previous section.

We list the building blocks or different entities that you can use to define your data model as follows:

Attribute	It is used to describe a property of an item type. You may use an attribute to search or locate an item.
Attribute group	It is a set or list of attributes that group together. It can be used for a list of attributes that is repeated for more than one item type.
Component	It is another word for an item type. It is used when you define the hierarchical tree structure between item types.

Root component	It is in the top or first level of the hierarchical tree. The root component only contains attributes of the item type that have single values.
Child component	It is in the second or lower level of the hierarchical tree. It is used when an attribute of an item type has multiple values.
Object	It is a data entity that is stored in Resource Manager. Library Server manages objects through items of particular item types.
MIME type	It is an Internet standard that is used to identify the type of object stored in Resource Manager. It is used when the document is displayed.
Item	It is an instance of an item type. It may contain one root component and zero or more child components.
Item type subset	It is a view of an item type that only contains certain properties or attributes of the item type.
Item classification	It is used to describe an item type when it has only attributes and no parts. This can be used to define a folder in the data model.
Resource item classification	It is used to describe an item type when it has attributes and an associated object stored in Resource Manager.
Document item classification	It is used to describe an item type that has attributes and multiple parts.
Document part item classification	It is used to define a new type of part that can be used as a part of a document item type.
Link	It is used to define the relationship between different item types. It is also used to define parent child relationships.
Foreign key	It is used to link an item type with an external table defined in the same DB2 database.
Reference attribute	It is a single-direction association between a root or child item type and another root item type.
Semantic type	It is an attribute that describes the behavior of an item type when used by a client application.
Versioning	It is used to keep previous instances of item type, a semantic type, a document, or a document part.

Index                                      It is used to identify an attribute of an item type that is searchable. This is done to optimize performance.

You can define these definitions by using the system administration client.

For more information, refer to the following publications:

- ▶ Chapter 7, “Modeling data in DB2 Content Manager” in *IBM DB2 Content Manager for z/OS V8.3: System Administration Guide*, SC27-1335
- ▶ Chapter 3, “Data modeling” in *Content Manager Implementation and Migration Cookbook*, SG24-7051

In the remaining section, we discuss each data modeling entity in detail, including the auto-linking feature.

## Auto-linking

You can use auto-linking to configure the association of an attribute or attribute group in one item type with an attribute or attribute group in another item type.

With auto-linking, attributes link associations between item types. When data is entered into an attribute or attribute group, it is also entered into the attribute of the associated item type.

Content Manager provides two default link types:

- ▶ *Folder contains* is used when you want to link the document contained in a folder with a physical folder.
- ▶ *Containment relationship* is used where the resource of each item is contained in the linked item folder, but it is still treated as a separate entity by Library Server.

**Note:** With Content Manager V8.3 and Fix Pack 6 of Content Manager V8.2, the system administration client no longer allows you to create auto-linking relationships with attributes that are not defined as required attributes. This is done to ensure that the auto-linked item type will be created properly.

To enable auto-linking:

1. On the auto-linking page of a new item type definition (see Example 2-2), select the **Only show available matching attributes and groups** check box. This ensures that only attributes and attribute groups at the same level are displayed.
2. Select an item type from the Item type to be linked to list. A list of attributes and attribute groups for that item type displays.

3. Select attributes or attribute groups from the Current item type list and Item type to be linked to list.
4. From the Link type list, select a link type to associate the attributes or attribute groups.
- Click **Add** to create a link set and add the attributes to the Associated attributes and groups list.

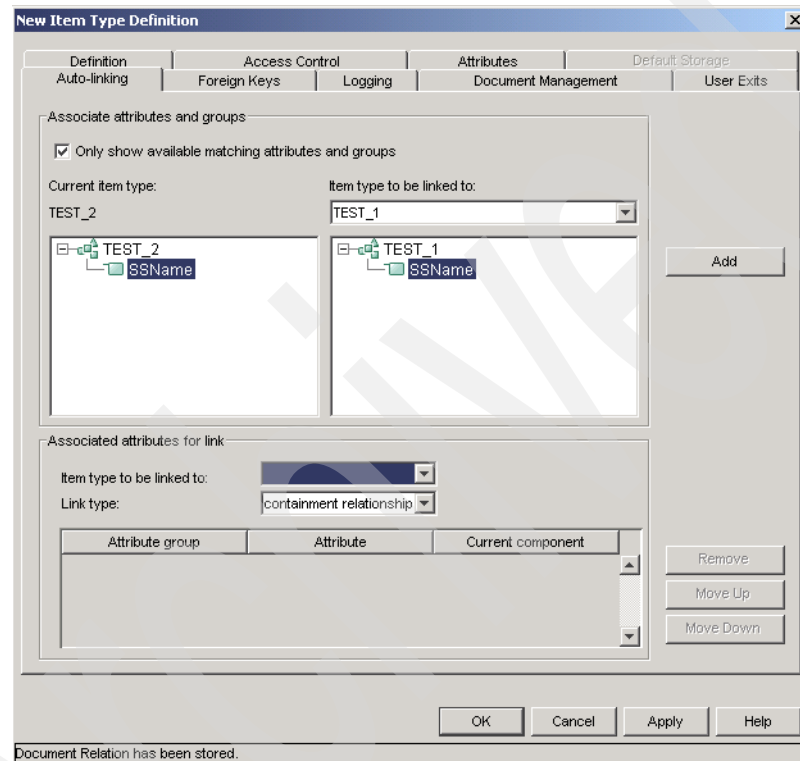


Figure 2-2 Create auto-linking

5. From the Item type linked to list under Associated attributes for link, select an item type. All attributes from this item type that are linked to the current item type display in the Associated attributes and groups list.

## Document parts

Document parts are used to define an item type when the item type is linked to a physical document that is stored in Resource Manager. This is done by adding one or more document parts to the item type definition of the document (see Figure 2-3).

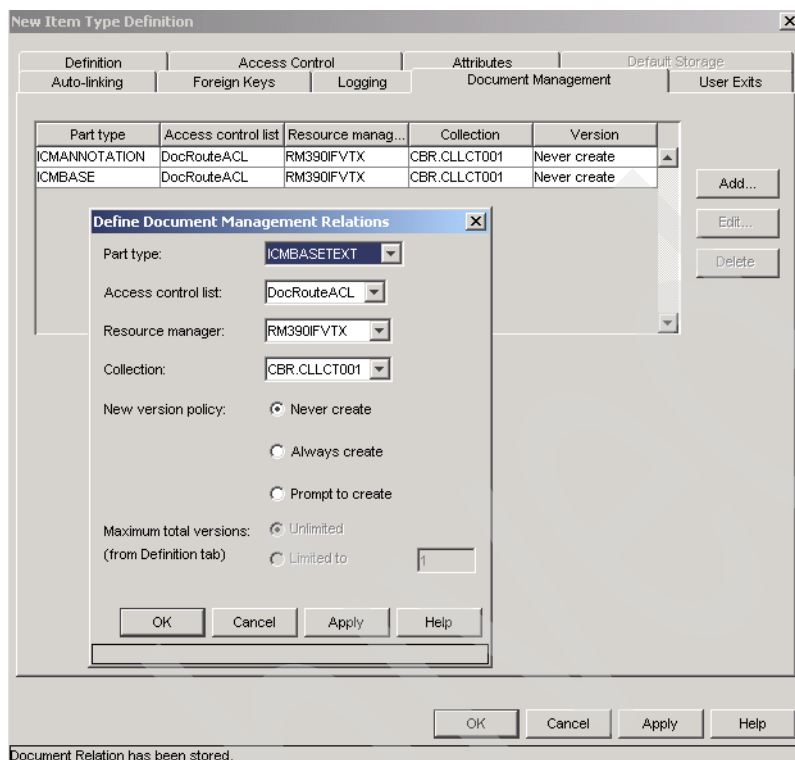


Figure 2-3 Adding document part in item type definition

Content Manager provides five predefined document parts that can be used:

- ICMANNOTATION It is used to add graphical annotations in the text of a document. Annotations are sticky notes or highlights that a user may add to a document after it is stored in Resource Manager. The Windows client or eClient can display the document with or without the annotations.
- ICMBASE It is used to indicate that a non-textual type of content will be stored in Resource Manager and that it will be linked to this item type.
- ICMBASETEXT It is used to indicate that the content of the document stored in Resource Manager is text searchable. This document part does not work when your Resource Manager is on z/OS.
- ICMNOTELOG It contains a log of information or comment entered by the users after the document is stored in Resource Manager.



ICMBASESTREAM It is used when you want to store streamed data such as video in Resource Manager.

Content Manager also provides an interface to define additional document parts. Note, these document parts are not supported by the standard Windows client or eClient.

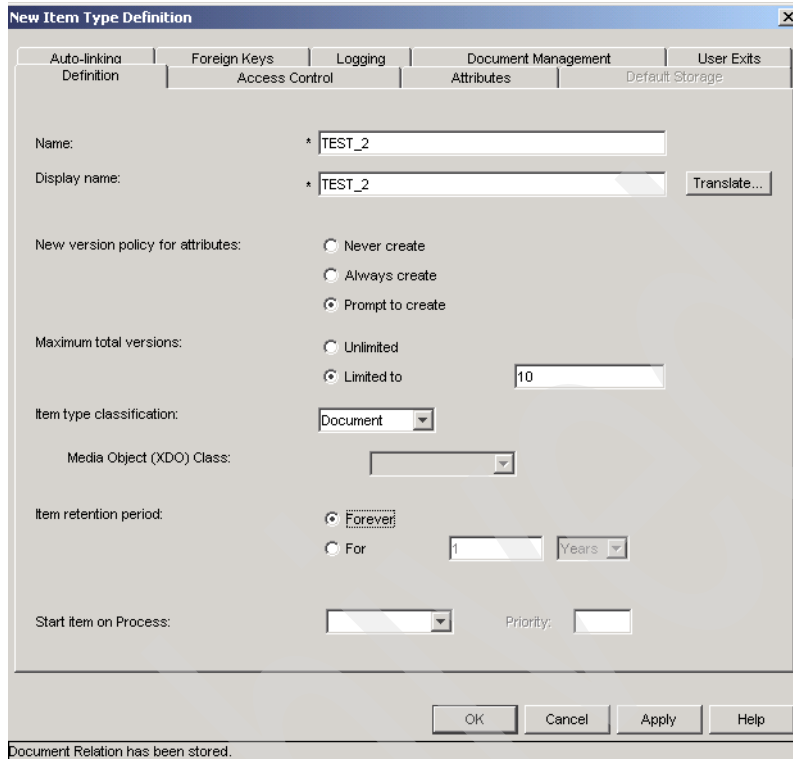
**Note:** You must add the ICMBASE document part to your item type if you want it to be displayed from the eClient.

## Versions

If your business requires that an original document be preserved and any modifications to it must be stored in a new version, Content Manager provides the following version policies that can be used when you create a new item type:

- ▶ *Never create* is used when you only store one version of an item. This will always be the newest version.
- ▶ *Always creates* is used when you want to keep a copy of an item type each time it is altered without the knowledge of the user.
- ▶ *Prompt to create* is used when you want to allow the user to decide when a new version should be created or if the current version should be overwritten.

Figure 2-4 shows the screen where you define versioning for attributes.



**New Item Type Definition**

Auto-linking | Foreign Keys | Logging | Document Management | User Exits  
 Definition | Access Control | Attributes | Default Storage

Name: \* TEST\_2

Display name: \* TEST\_2 Translate...

New version policy for attributes:  
☐ Never create  
☐ Always create  
☒ Prompt to create

Maximum total versions:  
☐ Unlimited  
☒ Limited to 10

Item type classification: Document

Media Object (XDO) Class:

Item retention period:  
☒ Forever  
☐ For 1 Years

Start item on Process: Priority:

OK Cancel Apply Help

Document Relation has been stored.

Figure 2-4 Define versioning

The maximum number of versions that you want to keep can be specified. In the case where the maximum number of versions has been reached, the oldest version will be deleted when a newer version is saved.

**Note:** You may not use the version policy “always create” or “prompt to create” if you use an attribute that is unique.

## MIME types

MIME is the abbreviation for Multipurpose Internet Mail Extension. It is an Internet standard for specifying the type of an object or document. It is also used to associate a specific type of object or document with a specific type of application. This association is used when users view an object or a document.

For example, you can associate PDF MIME type with Adobe Acrobat Reader. This association is normally done by the installation of Adobe Acrobat Reader. If the association is established, a user who opens a PDF file would automatically launch the Adobe Acrobat Reader which displays the file.

The Adobe Acrobat Internet plug-in can be used to enable products such as Microsoft® Internet Explorer to display PDF documents. In this case, the PDF documents can be displayed in the browser and you do not have to launch any application.

You can also link MIME types to applications in Content Manager. There is a list of predefined MIME types that are often used in a Content Manager implementation.

For the implementation of Content Manager, some of the considerations include:

- ▶ Do you have to expand the list of MIME types already defined in Content Manager?
- ▶ In the eClient configuration, you should specify if you want to launch an application to view the document or if you want to view it inside the browser.
- ▶ The eClient also gives you the ability to associate a Java applet with a MIME type. A Java applet is provided with eClient to view MO:DCA™ documents.
- ▶ In eClient, you also have the ability to transform a document from one type of format.

### Item type subsets

An item type subset is used to make a view of an item type. An example of this is when you have a item type called employee and the salary must not be displayed to all users.

In this case, you can create a view of a customer item type that includes all the attributes of the employee, except for the salary.

To create an item type subset, do the following:

1. Expand Data Modeling within the system administration client.
2. Expand an item type in the tree.
3. Right-click **Item Type Subsets** and click **New** to open the New Item Type Subset window (see Figure 2-5).
4. Fill the required fields.
5. Select an access control list for the item type subset.
6. Select the available attributes that you want users to view and click **Add**.
7. Use the Attribute filter for view fields so that users can only see attributes with certain values.
8. Click **OK** to create the item type subset.

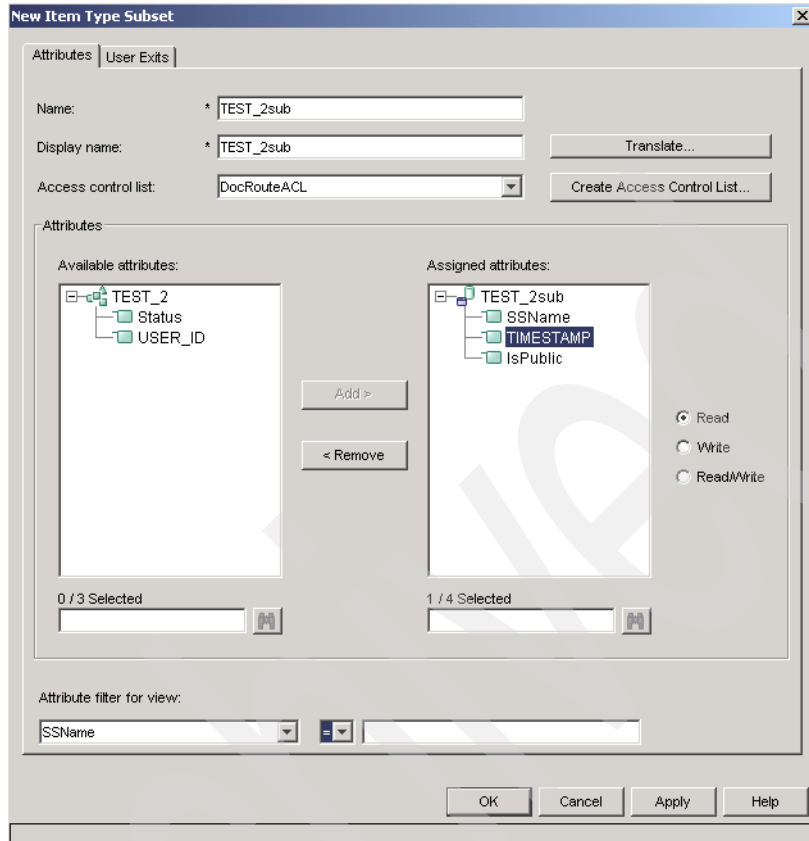


Figure 2-5 Creating item type subsets

## Semantic types

Semantic types are used to help client applications identify the semantics or behavior of an item. For example, you can have documents inside a folder. Both document and folder are semantic types. Content Manager provides the following predefined semantic types:

Annotation	Sticky notes or comments that are added to the document to highlight something after the document is stored in Content Manager.
Base	The content of the item stored in Content Manager. Some examples are image, text, and audio.
Container	This can be used to contain multiple items.

Document	A document is normally used when you have an item with more than one part. An example of a document is an item that has a base and an annotation.
Folder	A folder contains items that can be documents, or it contains other folders.
History	A history is a log of activities that is generated by an application. This semantic type was originally used by ImagePlus. Please note that you may not use this semantic type. It is only provided for migration purposes.
Note	This is a text log of information entered by the users.

You may extend the list of semantic types by defining your own.

## Database index

When you create an item type, Content Manager creates DB2 tables and views for that item type. When you search for a document that you previously imported, Content Manager accesses the item type tables to find the document.

Depending on the search criteria, it may take a while to return the information. To improve the performance of a search, you may create database indexes on searched attributes.

A database index is defined on an item type, and you select the attributes of the item type that you want to include in it. You may add multiple indexes on an item type. A database index may contain more than one attribute.

You add a database index in the case when you have an item type that contains a large number of items and the attribute you want to search on is not part of an index.

If no database index is defined on an attribute that is used in a search, DB2 performs a table scan. This is a very expensive operation for DB2, especially if the table has a large number of rows.

**Note:** An additional database index will require additional storage and it will also cause the addition or insertion of new items to be slower.

To create a database index, do the following:

1. Expand Data Modeling within the system administration client.
2. Select **Item Types**.
3. Expand the item type in which you are interested in the tree.

4. Select **Database Indexes**. The root and child components of the item type are displayed.
5. Right-click the component for which you want to create an index, and click **New** to open the New Database Index window (see Figure 2-6).
6. Fill the required fields.
7. In the Available attributes list, the attributes for the item type are displayed. Click **Add** to move them into the Assigned attributes list.
8. In the DB2 storage/retrieval field, specify the order for the attributes.
9. Click **OK** to create the database index.

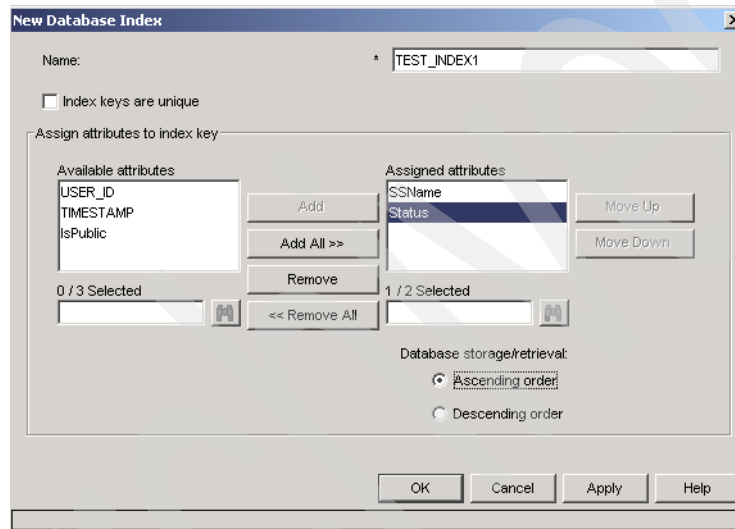


Figure 2-6 Create database index

## 2.4 Implement user access

Within the system administration client, you implement user access by defining:

- ▶ Users who are allowed to use the system and their passwords.
- ▶ User groups that define a set of users with common access control.
- ▶ Privileges that allow a user to access objects in a specific way.
- ▶ Privilege group which is a collection of privileges.
- ▶ Privilege set which is a collection of privileges that defines the user roles.
- ▶ Access control list which is a list of users or user groups and their associated privileges. It is used to protect access to objects in your system.

## 2.4.1 Users and user groups

In Content Manager, you have to define all the users who are going to use the system. You have an option to define and save the passwords of the users inside Content Manager or use system password from an external system.

In Chapter 5, “Security and exits” on page 111, you can find more information on how to configure Content Manager to use RACF or an external application to validate passwords. We also provide information on how to import users from RACF.

For every user in Content Manager, you have to define the following defaults:

- ▶ A default Resource Manager
- ▶ A default collection
- ▶ A default access control list

The defaults are used when a new object is created in Content Manager. For example, you need to specify on which Resource Manager you want to store a new document and which collection should be used.

You should group users into user groups. Always associate a user group to a privilege in Content Manager. Avoid assigning an individual user to a particular privilege.

## 2.4.2 Privileges, privilege groups, and privilege sets

A *privilege* in Content Manager is the right for a user to access an object in a specific way. Another way of describing this is to say that it gives a user the right to perform a specific action. The action can be related to the Windows client, eClient, or the system administration client.

Content Manager provides a list of privileges that may include some of the following:

- ▶ Client privileges
- ▶ Administrative privileges
- ▶ Item privileges

Content Manager gives you the ability to create your own privileges, but the default Windows client and eClient will not be able to act on it.

A *privilege group* is a collection of privileges. It is only used to make the administration of privilege sets easier.

A *privilege set* is a collection of privileges that defines a user role in your Content Manager system.

The maximum privilege set of a user indicates the maximum (or all) actions a user can perform. An example of this is that a user may have the privilege set:

```
ClientUserAllPrivs
```

This means that the user is allowed to perform all actions from the client. This is the maximum privilege set the user may perform. The user might be further restricted by the access control list that is specified to a specific item type.

### 2.4.3 Access control lists

Access control lists (ACL) are used in Content Manager to protect access to objects in your system. This includes the following:

- ▶ Objects or documents stored in Resource Managers
- ▶ Item types such as folders
- ▶ Workflow nodes such as a work basket
- ▶ Workflow processes
- ▶ Workflow work lists

An access control list is a list of users or user groups associated with a list of their privileges.

It is always best to assign user groups to an access control list instead of individual users.

Content Manager uses both the privileges of the user and the access control lists to check if a user may perform an action on an object. It first checks if the user has the privilege to perform a specific action, and then it checks the user's access control list to see if the user has access to the specific object. Both conditions have to be true before Content Manager system allows the user access to the object.

## 2.5 Implement workflow

There are two methods available in Content Manager to implement workflow. The first is called document routing and it provides an out-of-the-box solution with limited workflow capabilities.

The second method is to use advanced workflow with IBM DB2 Content Manager Information Integrator for Content and MQSeries® workflow. This is only available for Multiplatforms and not for z/OS. You have the option to install Content Manager system on z/OS and then connect to it from a machine with Information Integrator for Content installed that runs on Multiplatforms. This also



gives you the ability to integrate with MQSeries workflow. You can use MQSeries workflow to implement a complex workflow configuration.

For this IBM Redbook, we focus on the document routing. To implement it, you need to understand and document your business processes. You also need to understand the following building blocks or concepts of document routing and when and how to use it:

- ▶ Actions and action lists
- ▶ Workflow nodes
- ▶ Workflow processes
- ▶ Work lists
- ▶ Work packages

### 2.5.1 Actions and action lists

An *action* is used to specify how a user acts on a work package at a work node. Content Manager provides the following predefined actions:

- ▶ CMclient start on process: Starts a work package in a workflow process.
- ▶ CMclient remove from process: Removes a work package from a workflow process.
- ▶ CMclient change process: Removes a work package from one workflow process and starts on another.
- ▶ CMclient view process information: Views information about a work package in a workflow process.
- ▶ CMclient continue: Moves a work package from one step in the workflow process to the next step without specifying a route name.
- ▶ CMclient suspend: Temporarily suspends a work package in a workflow process.
- ▶ CMclient resume: Resumes a suspended work package in a workflow process.

Content Manager also gives you the ability to create new actions. You need to develop code for the function that will be performed by the new action in either a DLL for the Windows client or a JSP™ page for eClient.

You have to include an action in an action list to use it. An *action list* is a collection of actions that a user may perform on a work package at a work node.

## 2.5.2 Workflow nodes

A *work node* is a physical step in a workflow process where the work package has to wait for an action of an event. The action may be performed by a user or an application.

In Content Manager, there are three types of work nodes:

- ▶ Work basket

A *work basket* is a step in a workflow process where a work package has to wait for an action by a user or an application.

An action list may be provided to give the user different options on where to route the work package. If no action list is specified, the Windows client or eClient will only display the route names as actions.

A variable page is also provided to prompt the user for additional information.

- ▶ Collection point

A *collection point* is a step in a workflow process where a work package has to wait for additional item types or documents in a folder before proceeding.

A collection point has the same properties as a work basket. You also have to specify a resume list.

The resume list is a list of documents that have to arrive in a folder before the work package resumes in the work process.

You normally use a collection point in your business processes when you have to wait for additional documentation before you can complete a business task.

- ▶ Business application

You use a business application work node when you want to route a work package to an external application.

You have to develop the business application of the integration to the node yourself using the document routing user exits described in 4.3, “Document routing user exit routines” on page 80.

## 2.5.3 Workflow processes

A *workflow process* is a series of steps that a work package has to complete. Each step has to be defined by a work node or a virtual node. The most basic workflow process must have the following steps:

- ▶ Begin step
- ▶ One work node
- ▶ End step

The following is a list of virtual nodes:

- ▶ Start node: Indicates the start of a workflow process.
- ▶ Stop node: Indicates the end of a workflow process.
- ▶ Split node: Indicates the start of a parallel route in a workflow process.
- ▶ Join node: Indicates the end of a parallel route in a workflow process.
- ▶ Decision point: Routes a work package to a different work node depending on one of the following:
  - Information entered by the user in a variable defined on the work node.
  - Any of the properties of the work package.
  - Any of the attributes of the item that is being routed.
- ▶ Subprocess: Use a subprocess to include a predefined workflow process in your workflow process.

You can use the graphical workflow process builder tools to define the steps and the routes between the different steps for your workflow process. The steps in your workflow process are indicated by work nodes and virtual nodes.

You have to use a connector to link the processes with each other. A node can be linked to one or more nodes using a connector. The names of the connectors will be displayed as action names in the menus of the Windows client or eClient.

## 2.5.4 Work lists

A *work list* is a view or a filter of work packages that a user has to process. The work list may contain work packages from different work processes or work that waits for an action at different work nodes.

A user always receives work from a work list. You have to specify the workflow nodes and workflow processes that are filtered by the work list.

The work lists are then used to prioritize and organize the work that a user has to complete over many different processes.

## 2.5.5 Work packages

A user does not create a work package. A work package is an item that is started in a workflow process. A work package contains the information that is needed by the user to complete a workflow step at a specific work node.

Users may perform the following actions on a work package:

- ▶ Update or view the properties of a work package.

- ▶ Perform a get work on a work list. The work package will then be assigned to that user. The user can then perform the relevant business task to complete the workflow task.
- ▶ Route a work package from one work node to another work node.

The following publication provides more information on the components of Content Manager for both Multiplatforms and z/OS:

- ▶ *IBM DB2 Content Manager Enterprise Edition V8.3: Planning and Installing Your Content Management System*, GC27-1332

The following IBM Redbook provides more information about the implementation of Content Manager:

- ▶ *Content Manager Implementation and Migration Cookbook*, SG24-7051

## 2.6 Manage object storage

Objects in Content Manager for z/OS are stored in Resource Manager. Within Resource Manager, objects are grouped using collections, which are managed by the storage subsystem.

### 2.6.1 Storage management systems

There are two storage management systems available for z/OS, OAM, and Tivoli Storage Manager.

In Resource Manager for Multiplatforms, you also have to use a device manager and a storage class. This is not needed for Resource Manager for z/OS. There are also more store management systems available in the Multiplatforms environment.

You may connect Resource Manager that runs on UNIX or Windows to a Library Server that runs on z/OS.

### 2.6.2 Collections

You need to decide on the following before you can implement Resource Manager:

- ▶ What storage management system are you going to use for your Resource Manager?
- ▶ How many Resource Managers are you going to use?
- ▶ On what platform do you want the additional Resource Managers to run?

Once you decide on your storage management system and the configuration of your Resource Managers, you need to:

1. Define and establish the OAM/TSM collection name to be used in the SMS system.
2. Define a collection to Content Manager based upon that which is already successfully established in the SMS system.

In Content Manager, you have to link a default collection to each user and each item type.

In the Library Server configuration, you have to specify how to get the default collection of Resource Manager by choosing one of the following:

- ▶ Use the default collection of the item type
- ▶ Use the default collection of the user

If you choose the default collection of the item type, then when you store a new object or document in Resource Manager, it is stored in the default collection of the item type.

If you choose the default collection of the user, then when you store a new object or document in Resource Manager, the object or the document is stored in the default collection of the user.

### **2.6.3 Capacity planning**

Resource Manager uses the bulk of the disk space in Content Manager to store physical objects. You can estimate the storage by multiplying the average size of the objects with the total number of objects that you need to store.

You can group objects stored in Resource Manager by using collections. The collections in z/OS are managed by the storage management system.

Note that an OAM collection in z/OS cannot be greater than 64 GB when using DB2 segmented table spaces. When using partitioned table spaces, it cannot be greater than 16 TB for DB2 V7 and greater than 128 TB for DB2 V8.

### **2.6.4 Migration policies**

Migration policies are used to move objects stored in Resource Manager from one storage class to another. For example, you can do this by moving objects that are infrequently accessed to a cheaper medium.

In z/OS, you have to specify and manage your migration policies from the storage management system. In the Multiplatforms environment, you can do this with Content Manager.

## 2.6.5 Replication

You can use *replication* to move objects from a primary Resource Manager to a second Resource Manager. You should not use replication to replace your system backups, but you can use replication as a failover system for Resource Manager.

Replication can also be used as a faster method to restore a primary Resource Manager in case of a disaster or a disk failure.

You can also configure a Resource Manager on the Multiplatforms environment to act as LAN cache. This feature is not available on z/OS, but you can use it if you have a second Resource Manager on Multiplatforms.

The idea behind this is that objects or documents are temporarily stored on a second Resource Manager after they are fetched from the first Resource Manager. Any subsequent access to them will be from the second Resource Manager, which may be faster. This is normally done when the network access to the first Resource Manager is very slow or geographically remote.

## 2.7 Manage different domains

You can configure administrative domains to enable multiple users to administer Content Manager. This is normally done when you have multiple departments in your organization and you want to appoint an administrator for each department.

You will still need an administrator who is responsible for the entire Content Manager system. The first step is to enable administrative domains, from the system administration client, using the configuration settings.

**Note:** Please note that you cannot disable multiple domains once you have enabled them.

There are three predefined domains in Content Manager:

- ▶ SuperDomain: You must be under this domain to create new domains or assign components to a domain.
- ▶ PublicDomain: This is used for components that are used by all the users.

- ▶ **DefaultDomain:** This is only used for administrative purposes when a new user or component is created.

You can create new domains, one for each department, and give administrative access to the department's administrator in each domain.

Once you have created the domains, you have to assign components to them. The following components can be assigned to a new domain:

- ▶ Users
- ▶ User groups
- ▶ Collections
- ▶ Resource Manager

A component can only be linked to one domain at a time. This includes the public domain. This means that a component is either available to all users or only the users of a specific domain.

## 2.8 Provide Web access

Content Manager for z/OS does not provide out-of-the-box Web access. You have two options to get Web access:

- ▶ You can use IBM DB2 Content Manager Information Integrator for Content for Multiplatforms in conjunction with the Content Manager eClient for Multiplatforms, and connect it to the Content Manager system running on z/OS. This gives you Web access via the Multiplatforms environment.
- ▶ You can develop your own Web application using Content Manager Toolkit for z/OS and an application server such as IBM WebSphere Application Server for z/OS.

In 3.8, "Using Content Manager Toolkit for z/OS from a Web environment" on page 68, we present an example of a servlet that can be used from a Web environment.

You can get more information on this topic from the following IBM Redbooks:

- ▶ *eClient 101 Customization and Integration*, SG24-6964
- ▶ *Implementing Web Applications with Content Manager Information Integrator for Content and OnDemand Web Enablement Kit*, SG24-6338
- ▶ *WebSphere Portal Server and DB2 Information Integrator: A Synergistic Solution*, SG24-6433

## 2.9 Backup and recovery

A well-defined backup and recovery strategy is crucial for the availability of a Content Management system. A Content Manager for z/OS system basically consists of DB2, OAM or TSM, and HTTP. We must have a backup strategy that covers all of these software products. Remember that our backup procedure should not only cover the Content Manager stored data, but it should also cover all of the configuration files and load libraries, too.

You must ensure that, in the recovery process, all the data is restored at the same point-in-time. Otherwise, we will have data-inconsistency problems.

Content Manager has an asynchronous recovery process that detects discrepancies between Resource Manager and Library Server.

For more information, refer to the following IBM Redbooks and publication:

- ▶ *Content Manager Backup Recovery and High Availability Strategies, Options, and Procedures*, SG24-7063
- ▶ *Disaster Recovery with DB2 UDB for z/OS*, SG24-6370
- ▶ *IBM DB2 Content Manager for z/OS: Planning and Installing Your Content Management System*, GC18-7698

## 2.10 Performance tuning

Performance tuning for a Content Manager system is not simply modifying some parameters after the Content Manager installation. You should follow a methodology for building and maintaining a tuned Content Manager system. The Content Manager performance team recommends the following methodology:

- ▶ Document the desired system topology, projected workload, and the setting of measurable performance objectives.
- ▶ Perform initial capacity planning and sizing activities.
- ▶ Understand design choices and their performance trade off.
- ▶ Start the performance tuning.
- ▶ Monitor the system to maintain the performance.

The following hints and tips could help in performance tuning:

- ▶ Have adequate system resources.
- ▶ Monitor system usage.
- ▶ Reduce the number of extents for any DB2 table space and index space.



- ▶ Adjust DB2 buffer pools.
- ▶ Run DB2 utilities reorg and runstats frequently.
- ▶ Create item type database indexes for the most searched attributes.
- ▶ Avoid searches using wild cards, such as “ABC%” or “ABC\*” or \*.
- ▶ If an application searches against a non-indexed attribute, determine if the search is really necessary. If it is, you may consider creating a new database index. Otherwise, eliminate or work around this search pattern with the application.

For more information, refer to the following IBM Redbooks and manuals:

- ▶ *Performance Tuning for Content Manager*, SG24-6949.  
This redbook addresses Content Manager for Multiplatforms; but it has many recommendations that also apply for z/OS.
- ▶ Content Manager white paper *Performance Tuning Guide* at:  
<http://www.ibm.com/support/docview.wss?uid=swg27003894>
- ▶ *IBM DB2 Content Manager for z/OS V8.3: System Administration Guide*, SC27-1335

## 2.11 Additional products that use Content Manager

IBM provides a number of products that use the functionality of Content Manager to implement a new out-of-the-box business solution.

The following are some examples of products that exploit Content Manager:

- ▶ IBM DB2 Records Manager  
Records Manager is a tool for applying formal records management policies to your documents by managing the life cycle of them.  
You can get more information on Records Manager at:  
<http://www.ibm.com/software/data/cm/cmgr/rm/>
- ▶ IBM DB2 CommonStore for Exchange server  
CommonStore for Exchange provides an integrated electronic archiving solution for Microsoft Exchange server.  
You can get more information on CommonStore for Exchange at:  
<http://www.ibm.com/software/data/commonstore/exchange/>
- ▶ IBM DB2 CommonStore for Lotus® Domino®

CommonStore for Lotus Domino provides an integrated electronic archiving solution for IBM Lotus Notes®.

You can get more information on CommonStore for Lotus Domino at:

<http://www.ibm.com/software/data/commonstore/lotus/>

► IBM DB2 CommonStore for SAP

CommonStore for SAP provides an integrated electronic archiving solution for SAP applications.

You can get more information on CommonStore for SAP at:

<http://www.ibm.com/software/data/commonstore/sap/>

► IBM DB2 Document Manager

Document Manager makes it easy to keep track of critical business information by managing the life cycles of your documents.

You can get more information on Document Manager at:

<http://www.ibm.com/software/data/cm/docmgr/>

# Content Manager Toolkit for z/OS

In this chapter, we describe IBM DB2 Content Manager Toolkit for z/OS V8.3, also known as host APIs in the z/OS environment. Content Manager Toolkit is new to Content Manager V8.3 for z/OS. Previously, Content Manager Toolkit was available only in the multiplatform environment.

We cover the following topics:

- ▶ Configuring the z/OS development environment
- ▶ Content Manager Toolkit for z/OS interfaces overview
- ▶ Compiling and running the sample programs
- ▶ Sample ICM programs provided on z/OS
- ▶ Sample Java bean programs provided on z/OS
- ▶ Using Content Manager Toolkit for z/OS from a batch environment
- ▶ Using Content Manager Toolkit for z/OS from a Web environment

## 3.1 Introduction

Content Manager for z/OS V8.3 comes with Content Manager Toolkit for z/OS (also known as host APIs) that enables you to customize your Content Manager solution or create your own Content Manager applications.

By using the Content Manager Toolkit for z/OS, you can perform tasks such as creating, updating, and deleting a data model, creating and processing workflow, and providing user security management.

You call the APIs from within your Java application. If you need to call them from a different development environment, wrap the Java code that uses the Content Manager Toolkit. The wrapped code can then be called from a different environment.

The applications you create with Content Manager Toolkit for z/OS can be called from a command line, a batch process, or from the Web, depending on how you create them.

## 3.2 Configuring the z/OS development environment

In this section, we discuss how you can configure your z/OS development environment in order to develop applications using Content Manager Toolkit for z/OS.

Make sure you have the prerequisite software for Content Manager Toolkit for z/OS. Refer to the program directory provided with Content Manager Toolkit for z/OS. Once you install the APIs, you also need to configure the environment variables in order to ensure that your application can work with the APIs.

### Installation

This topic is described in Chapter 6, “Installation” on page 153.

### 3.2.1 Configuring environment variables

You need to configure environment variables in z/OS UNIX System Services *for each user* who wants to do development or use Content Manager Toolkit for z/OS. These environment variables include:

- ▶ PATH
- ▶ STEPLIB
- ▶ CLASSPATH

## PATH

Add the following paths to your PATH environment variable:

```
/bin  
/usr/lpp/java/J1.4/bin  
/usr/lpp/db2/db2c/jcc/bin
```

**Note:** You also need the following paths in your PATH environment variable:

```
/usr/lpp/icm/V8R3M0/cmgmt  
/usr/lpp/icm/V8R3M0/cmgmt/connectors  
/usr/lpp/icm/V8R3M0/lib
```

They are automatically set by running `/usr/lpp/icm/V8R3M0/bin/cmbenv81.sh`.

## STEPLIB

Add the following statements to your STEPLIB environment variable:

```
DB2C.DSNEXIT:DB2C.DSNLOAD  
DB2C.DSNLOAD2
```

## CLASSPATH

Add the following paths to your CLASSPATH environment variable:

```
/usr/lpp/db2/db2c/jdbc/jcc/classes/sqlj.zip  
/usr/lpp/db2/db2c/jdbc/jcc/classes/db2jcc.jar  
/usr/lpp/db2/db2c/jdbc/jcc/classes/db2jcc_license_cisuz.jar  
/usr/lpp/db2/db2c/jdbc/jcc/classes/db2jcc_javax.jar
```

### 3.2.2 Executing Content Manager Toolkit's environment script

After you set PATH, STEPLIB, and CLASSPATH environment variables as discussed in the previous section, you also need to execute the following script to configure additional environment variables and settings before you can compile or run any of the sample programs:

```
. /usr/lpp/icm/V8R3M0/bin/cmbenv81.sh
```

**Note:** The version of Content Manager Toolkit for z/OS is not related to the name of the environment script.

## 3.3 Content Manager Toolkit for z/OS interfaces overview

In this section, we explain the different interfaces of Content Manager Toolkit for z/OS. We also provide information on when to use each interface and its availability on the z/OS environment.

### 3.3.1 Java interface

Content Manager Toolkit *Java interface* ( also known as ICM Java interface) is for working with Content Manager only. It is used to manipulate Content Manager on a lower level of abstraction.

This interface has a very rich set of APIs to enable you to create, update, or delete any part of the data model or configuration of Content Manager. It can also be used to create, update, or delete any data stored in Content Manager.

The data stored in a Content Manager database includes folders, documents, document parts, and the attributes of these items.

The ICM Java toolkit interface is available in Content Manager Toolkit for z/OS.

### 3.3.2 Non-visual bean interface

The *non-visual bean interface* is implemented using the Content Manager Toolkit Java interface and it uses a higher level of abstraction.

This interface is very easy to use but it has less functionality than the ICM Java interface. You normally use it to develop code that has to be generic.

**Note:** The non-visual bean interface can be used to connect to other types of repositories, such as OnDemand and ImagePlus. The APIs are available for z/OS.

### 3.3.3 Visual bean interface

The *visual bean interface* is used to display visual components, such as for viewing a document or its annotations.

The visual bean interface is not available in Content Manager Toolkit for z/OS.

If you want to use it, you need to develop your application in the Multiplatform environment, using IBM DB2 Information Integrator for Content.

### 3.3.4 Content Manager Toolkit for z/OS package imports

You need to add the following import statements when you develop a program that uses the ICM interface:

```
import com.ibm.mm.sdk.common.*;
import com.ibm.mm.sdk.server.*;
```

For the bean interface, you need to add the import statement:

```
import com.ibm.mm.beans.*;
```

### 3.3.5 Error or exception handling

Content Manager Toolkit for z/OS contains classes that perform error or exception handling. Example 3-1 contains example code for handling errors or exceptions when you are working with the ICM interface.

*Example 3-1 Example code for error and exception handling with ICM APIs*

---

```
try {
    ... HOST API Operations ...
} catch (DKException dke) {
    System.out.println("DK Exception occurred: ");
    System.out.println(" Name: " + dke.name());
    System.out.println(" Message: " + dke.getMessage());
    System.out.println(" Message ID: " + dke.getErrorId());
    System.out.println(" Error State: " + dke.errorState());
    System.out.println(" Error Code: " + dke.errorCode());
    dke.printStackTrace();
} catch (Exception e) {
    System.out.println("Exception occurred: ");
    System.out.println(" Name: " + e.getClass().getName());
    System.out.println(" Message: " + e.getMessage());
    e.printStackTrace();
}
```

---

Example 3-2 contains a code example for handling errors or exceptions when you are working with the bean interface:

*Example 3-2 Code example on error and exception handling with bean interface*

---

```
try {
    ... HOST API Operations ...
} catch (CMBException ce) {
    System.out.println("CMB Exception occurred: ");
    System.out.println(" Name: " + ce.name());
    System.out.println(" Message: " + ce.getMessage());
    System.out.println(" Datastore Error ID: " + ce.getDKErrorId());
    System.out.println(" Datastore Error Code: " + ce.getDKErrorCode());
    System.out.println(" Datastore Error State: " + ce.getDKErrorState());
    System.out.println(" Error ID: " + ce.getErrorId());
    System.out.println(" Error Data: " + ce.getErrorData());
    System.out.println(" Error Extra Data: " + ce.getErrorExtraData());
    dke.printStackTrace();
} catch (Exception e) {
    System.out.println("Exception occurred: ");
}
```

```
System.out.println(" Name: " + e.getClass().getName());
System.out.println(" Message: " + e.getMessage());
e.printStackTrace();
}
```

---

## 3.4 Compiling and running the sample programs

In this section, we discuss how you can compile and run the sample programs.

The sample programs must be compiled under the z/OS UNIX System Services environment.

The PATH, CLASSPATH, and STEPLIB environment variables have to be defined in the .profile file of the user who compiles the Java sample programs. If you have not configured these variables as discussed in the previous section, refer to 3.2.1, “Configuring environment variables” on page 46.

You also need to execute Content Manager Toolkit for z/OS’ environment script file before you can compile or run the examples. If you have not done so, refer to 3.2.2, “Executing Content Manager Toolkit’s environment script” on page 47.

### 3.4.1 Compiling and executing ICM sample programs

All the sample ICM programs files can be found in the path:

```
/usr/lpp/icm/V8R3M0/samples/java/icm
```

The sample programs use a specific data model and they access items loaded in this data model. In this section, we explain how to compile and execute the ICM sample programs. In the next section, we explain how to load the data model and how to set up a testing environment.

The compilation and execution of all the ICM sample programs are very similar. For this reason, we show you how to compile and run only one sample program, the SConnectDisconnectICM example.

The SConnectDisconnectICM example shows you how to connect and disconnect from the Content Manager Library Server for z/OS.

Example 3-3 is the example code containing the main logic of the example.

*Example 3-3 SConnectDisconnectICM example code*

---

```
DKDatastoreICM dsICM = new DKDatastoreICM();
dsICM.connect(database,userId,password,connectionString);
dsICM.disconnect();
```



```
dsICM.destroy();
```

---

On the first line of the Example 3-3, a datastore is created. On the second line, a connection is established between the datastore and Content Manager Library Server. On the third line, the connection is closed. On the last line, the datastore is destroyed.

The following parameters are required to make a connection to Content Manager Library Server:

- ▶ Content Manager Library Server database name
- ▶ User ID defined for Content Manager database
- ▶ Password for the provided user ID
- ▶ Additional connection parameters that you want to specify.

You may use an empty string if you have no connection parameters.

To compile and run this example, do the following:

1. Enter the z/OS UNIX System Services from TSO/E with the command:

```
tso omvs
```

2. In the z/OS UNIX System Services environment, execute Content Manager Toolkit for z/OS' environment script:

```
. /usr/lpp/icm/V8R3M0/bin/cmbenv81.sh
```

**Important:** All commands in z/OS UNIX System Services are case-sensitive.

3. Navigate to the ICM path with the following command:

```
cd /usr/lpp/icm/V8R3M0/samples/java/icm
```

4. Compile the sample program:

```
javac SConnectDisconnectICM.java
```

5. Execute the sample program:

```
java SConnectDisconnectICM database userid password
```

Replace the database, userid, and password with your own values.

Example 3-4 presents the output of the sample program.

*Example 3-4 Output of SConnectDisconnectICM example*

---

```
=====
IBM DB2 Content Manager                      v8.3
Sample Program: SConnectDisconnectICM
-----
```

```

Database: NQADB2C
UserName: ifvti
=====
Connecting to datastore (Database 'NQADB2C', UserName 'ifvti')...
Connected to datastore (Database 'NQADB2C', UserName 'ifvti').
Disconnecting from datastore & destroying reference...
Disconnected from datastore & destroying reference.

=====
Sample program completed.
=====

```

---

### 3.4.2 Compiling and executing non-visual bean samples

The compilation and execution of all the non-visual bean samples are also very similar. In this case, we work with one sample program, TConnectionPool, to show you how to compile and execute non-visual bean samples.

The TConnectionPool bean is used to connect and disconnect from Content Manager Library Server for z/OS. The difference between this example and the previous example, SConnectDisconnectICM, is that we use a connection pool in this example.

*Connection pools* are used to optimize resource usage when there are a lot of connections being made to the same system.

Example 3-5 is the example code containing the main logic of the example.

*Example 3-5 TConnectionPool example code*

```

CMBConnectionPool connectionPool = new CMBConnectionPool();
connectionPool.setDsType("ICM");
connectionPool.setServerName(server);
connectionPool.setConnectString("");
CMBConnection connection = connectionPool.getConnection(userid, password);
connectionPool.freeConnection(connection);

```

---

On the first line of the Example 3-5, a connection pool is created. The next three lines set the properties of the connection pool. On the fifth line, a new connection is created to Content Manager Library Server. On the last line, the connection is freed (closed).

**Note:** You have to specify the datastore type when you use the bean interface. "ICM" is used to indicate that you want to connect to a Content Manager database.

To compile and run this example, you need to perform the following steps:

1. Enter the z/OS UNIX System Services from TSO/E with the command:

```
tso omvs
```

2. In the z/OS UNIX System Services environment, execute the API environment script:

```
. /usr/lpp/icm/V8R3M0/bin/cmbenv81.sh
```

3. Navigate to the beans path:

```
cd /usr/lpp/icm/V8R3M0/samples/java/beans
```

4. Compile the sample program:

```
javac TConnectionPool.java
```

5. Execute the sample program:

```
java TConnectionPool database userid password
```

Replace the database, userid, and password with your own values.

Example 3-6 presents the output of the sample program.

*Example 3-6 Output of TConnectionPool example*

---

```
Connection obtained.
```

```
Entities: temp doc1 doc2 doc3 doc4 doc5 doc6 NOINDEX CMDRFOLDERS
```

```
Connection freed.
```

```
Connection obtained.
```

```
Entities: temp doc1 doc2 doc3 doc4 doc5 doc6 NOINDEX CMDRFOLDERS
```

```
Connection freed.
```

```
Connection obtained.
```

```
Entities: temp doc1 doc2 doc3 doc4 doc5 doc6 NOINDEX CMDRFOLDERS
```

```
Connection freed.
```

```
Connection obtained.
```

```
Entities: temp doc1 doc2 doc3 doc4 doc5 doc6 NOINDEX CMDRFOLDERS
```

```
Connection freed.
```

```
Connection obtained.
```

```
Entities: temp doc1 doc2 doc3 doc4 doc5 doc6 NOINDEX CMDRFOLDERS
```

```
Connection freed.
```

---

## 3.5 Sample ICM programs provided on z/OS

In this section, we discuss the sample ICM programs provided on z/OS. The ICM sample programs are developed using the Java programming language.

Some of the examples are not related to Content Manager for z/OS. For example, text searching is not available on Content Manager for z/OS.

There are also several examples that show you how to modify the data model and how to set the configuration of Library Server and Resource Manager. Since XML import and export functions have been added to Content Manager V8.3, the need for these APIs has decreased.

In this section, it is not our intent to discuss all the sample programs in detail. We list only the samples that are relevant to Content Manager for z/OS, give a short description for each example, and highlight the APIs introduced by each sample.

### **Available resources for Content Manager Toolkit for z/OS**

Please note that the ICM Java interface for Content Manager Toolkit for z/OS and Multiplatforms is exactly the same. You may use the documentation available for Multiplatforms to get detailed information on the usage of the classes defined in the Content Manager Toolkit for z/OS.

The information center is the best place to get started. You can install and use the information center locally on a single workstation. It is also available on the Internet at:

<http://publib.boulder.ibm.com/infocenter/cm83>

You may also use the following reference publications for more information:

- ▶ *DB2 Content Manager Enterprise Edition V8.3: Application Programming Guide*, SC27-1347

This manual gives you a very detailed description of Content Manager Toolkit.

- ▶ *IBM DB2 Content Manager Enterprise Edition V8.3: Application Programming Reference*, SC27-1348

The reference contains details about Content Manager and Information Integrator for Content APIs and classes. It should be used in conjunction with the Application Programming Guide mentioned above.

- ▶ *IBM DB2 Content Manager for z/OS V8.3: Messages and Codes*, SC27-1349

This manual provides the errors and messages returned by Content Manager Toolkit for z/OS.

## **3.5.1 Preparing the ICM sample programs**

To prepare the sample programs, you need to compile all the Java programs that are used by the sample programs and define the data model used by the sample programs.

The Java programs that need to be compiled are:

SConnectDisconnectICM.java  
SAttributeDefinitionCreationICM.java

```
SAttributeGroupDefCreationICM.java
SAttributeGroupDefRetrievalICM.java
SItemTypeCreationICM.java
SLinkTypeDefinitionCreationICM.java
SReferenceAttrDefCreationICM.java
SResourceMgrDefCreationICM.java
SResourceMgrDefSetDefaultICM.java
SSMSCollectionDefSetDefaultICM.java
```

The next step is to define the data model used by the sample programs. To do this, you need to modify the file `SSampleModelBuildICM.java` by changing the following values in the source code of the program:

```
String rmUserName      = "icmadmin";
String rmPassword      = "password";
String rmReferenceName = "MY_RM";
short  rmPlatform      = DKConstantICM.RM_PLATFORM_390;
String rmINetAddress   = "taj.stl.ibm.com";
short  rmPort          = 80;
short  rmAccessType     = DKConstantICM.RM_ACESSTYPE_HTTP;
String rmAccessData     = "/icmrm/ICMResourceManager";
short  rmTokenDuration = 0;
```

The first two values are the administrative user ID and password of your Resource Manager and the third line contains the name of your Resource Manager.

On the fifth line, you need to specify the host name of the computer running the Resource Manager.

Lines six to eight are related to the HTTP server of the Resource Manager. You need to modify these lines to indicate the port number and the Web site of the Resource Manager according to your setup.

Once you have modified `SSampleModelBuildICM.java`, you need to compile and execute the sample program by using the commands:

```
javac SSampleModelBuildICM.java
java SSampleModelBuildICM database username password
```

where the database is the name of the Library Server database and the username and password are the administrative user and password. Replace these values according to your system setup.

The program `SSampleModelBuildICM` configures both the Library Server and the Resource Manager by creating a data model and injecting data in it.

### 3.5.2 Transaction sample: STransactionsICM.java

The STransactionsICM example shows you how to implement transaction control. It shows you how to start, commit, and roll back a transaction.

You can execute the sample with the following command:

```
java STransactionsICM database username password
```

### 3.5.3 Item creation sample: SItemCreationICM.java

The SItemCreationICM example shows you how to create items by setting the item properties, semantic types, namespaces, attributes, and attribute groups. It also shows you how to create child components, multivalued attributes, and reference attributes.

You can execute the sample with the following command:

```
java SItemCreationICM database username password
```

### 3.5.4 Item deletion sample: SItemDeletionICM.java

The SItemDeletionICM example shows you how to delete items. It also shows you how to delete all items of an item type.

You can execute the sample with the following command:

```
java SItemDeletionICM database username password
```

### 3.5.5 Item retrieval sample: SItemRetrievalICM.java

The SItemRetrievalICM example shows you how to retrieve the items, child components, and child components with multivalued attributes.

It also shows you how to use the retrieval options and how to access attributes programmatically using the component definition.

You can execute the sample with the following command:

```
java SItemRetrievalICM database username password
```

### 3.5.6 Item modification sample: SItemUpdateICM.java

The SItemUpdateICM example demonstrates to you how to:

- ▶ Check an item in and out.
- ▶ Modify an item.
- ▶ Add values to multivalued attributes or child components.

- ▶ Remove values to multivalue attributes or child components.
- ▶ Update or save items.
- ▶ Modify options.

You can execute the sample with the following command:

```
java SItemUpdateCM database username password
```

### 3.5.7 Folder manipulation sample: SFolderICM.java

The SFolderICM example shows you how to manipulate folders in Content Manager. It demonstrates how to perform the following actions on a folder:

- ▶ Create a folder.
- ▶ Add items to a folder.
- ▶ Retrieve a folder.
- ▶ Retrieve the contents of a folder.
- ▶ Search or query folder contents.

You can execute the sample with the following command:

```
java SFolderICM database username password
```

### 3.5.8 Link manipulation sample: SLinkslCM.java

The SLinkslCM example shows you how to create, retrieve, modify, delete, or print links.

You can execute the sample with the following command:

```
java SLinkICM database username password
```

### 3.5.9 Search sample: SSearchICM.java

The SSearchICM example shows you how to perform the following tasks:

- ▶ Search or query the stored content.
- ▶ Perform a parametric search of items.
- ▶ Perform a text search of items.
- ▶ Combined search.
- ▶ Search by using the execute method.
- ▶ Search by using the evaluate method.
- ▶ Search with a callback method.
- ▶ Access and use the search results.
- ▶ Manipulate results collection.
- ▶ Use the result set cursor.
- ▶ Use the search options.
- ▶ Search and query string syntax.

You can execute the sample with the following command:

```
java SSearchICM database username password
```

### 3.5.10 Document routing sample: SDocRoutingListingICM.java

The SDocRoutingListingICM example shows you how to perform the following document routing related tasks:

- ▶ List all the routing processes.
- ▶ List all the work modes.
- ▶ List all the work lists.
- ▶ Count the number of packages currently in a work list.
- ▶ Obtain all work packages in a work list.
- ▶ Retrieve and list the work package information.
- ▶ Print the work package information.

You can execute the sample with the following command:

```
java SDocRoutingListingICM database username password
```

### 3.5.11 Deleting sample data

The SSampleDeleteICM example deletes the items and item types used by the sample programs.

You can execute the example program with the following command:

```
java SSampleModelDeleteICM database username password
```

## 3.6 Sample Java bean programs provided on z/OS

In this section, we describe the sample Java bean programs shipped with Content Manager Toolkit for z/OS.

You do need a data model to run the examples.

The examples differ from the ICM samples because these examples do not use a sample data model. They are more generic and can be used with any data model that you may define. For example, you can use the data model that was created by the DK APIs in the previous section.

The parameters of these examples are more complex than sample ICM programs. You need to give the names of the entity or item types and attributes on which you want the actions to be performed.



Substitute the following when you use the sample programs:

- ▶ **dstype:** Use ICM.
- ▶ **userid:** Use the administrator ID for the Content Manager Library Server.
- ▶ **password:** Use the password of the Content Manager administrator ID.
- ▶ **server:** Use the Content Manager DB2 database name.
- ▶ **linktype:** Use contains.

### 3.6.1 Link item sample: TAddLink.java

The TAddLink example shows you how to add a link between one item and another item or items.

You can use the example by executing the following command:

```
java TAddLink dstype server userid password linktype entity condition / entity  
con
```

where the first entity is a source item type (for example, B00K), and condition is a search string on the source entity (for example, @Title like \"RED B00K\").

The second entity is the destination entity and con is the condition of the destination entity. A link is then created between the source and destination entities on the specified conditions.

### 3.6.2 Folder creation sample: TCreateFolder.java

The TCreateFolder example shows you how to create a folder on the server.

You can use the example by executing the following command:

```
java TCreateFolder dstype server userid password entity attributes
```

where entity is any item type (for example, B00K), and attributes are a list of variables-equal-value pairs (for example, author='BOB' timestamp=timestamp).

### 3.6.3 Item creation sample: TCreateItem.java

The TCreateItem example shows you how to create an item with no content on the Library Server.

You can use the example by executing the following command:

```
java TCreateItem dstype server userid password entity attributes
```

where entity is any item type (for example, B00K), and attributes are a list of variables-equal-value pairs (for example, fname='minh2' AND ssn='123-64-1234').

### 3.6.4 Deletion sample: TDelete.java

The TDelete example shows you how to delete all items matching a search pattern.

You can use the example by executing the following command:

```
java TDelete dstype server userid password entity condition
```

where entity is any item type (for example, BOOK), and condition is a search string (for example, @Title like \"RED BOOK\").

### 3.6.5 Attribute update sample: TEditAttributes.java

The TEditAttributes example shows you how to edit the attributes of one or more items. The list of items is chosen by searching with a specific condition and then updating the values of the specified attributes.

You can use the example by executing the following command:

```
java TEditAttributes dstype server userid password entity condition attributes
```

where entity is any item type (for example, BOOK), and condition is a search string (for example, @Title like \"RED BOOK\").

The attributes are a list of variables-equal-value pairs that indicate the name of the attributes of the item that have to be changed and the new values (for example, LastName='Smith' FirstName='John').

The attribute list may also contain subattributes (for example, 'Author.LastName'='Jones').

### 3.6.6 Entity listing sample: TListEntities.java

The TListEntities example shows you how to list all entities and their attributes defined on a Content Manager Library Server.

You can use the example by executing the following command:

```
java TListEntities dstype server userid password entity
```

where entity is any item type (for example, BOOK). In this case, detailed information will be displayed about the entity.

Details of all the entities defined in the Library Server can be displayed if you do not specify the entity parameter.

### 3.6.7 Link listing sample: TListLinks.java

The TListLinks example shows you how to display the links to and from an item or items. Both inbound and outbound links are displayed.

You can use the example by executing the following command:

```
java TListLinks dstype server userid password linktype entity condition
```

where entity is any item type (for example, BOOK), and condition is a search string (for example, @Title like \"RED BOOK\").

### 3.6.8 Search sample: TSearch.java

The TSearch example shows you how to perform a search and display details on the items found that include the attributes, child components, folder contents, and document parts.

You can use the example by executing the following command:

```
java TSearch dstype server userid password entity condition
```

where entity is any item type (for example, BOOK), and condition is a search string (for example, @Title like \"RED BOOK\").

### 3.6.9 Link removal sample: TRemoveLink.java

The TRemoveLink example shows you how to remove the link between two items.

You can use the example by executing the following command:

```
java TRemoveLink dstype server userid password link entity condition / entity  
con
```

where the first entity is a source item type (for example, BOOK), and condition is a search string on the source entity (for example, @Title like \"RED BOOK\").

The second entity is the destination entity and con is the condition of the destination entity. A link is removed between the source and destination entities on the specified conditions.

## 3.7 Using Content Manager Toolkit for z/OS from a batch environment

In this section, we explain how to use Content Manager Toolkit for z/OS from a batch environment or a batch process in z/OS.

A batch process normally has the following components:

- ▶ JCL that contains the steps in your batch process
- ▶ Input data that is required by the business logic of your batch process
- ▶ Program code that executes the business logic of your batch process
- ▶ Output data that contains the result of your batch process

The APIs from Content Manager Toolkit for z/OS are called from z/OS UNIX System Services or from stand alone Enterprise COBOL.

This means that you have to develop a Java program to call the relevant APIs. Your business logic and main processing loop have to be wrapped inside this Java program. The Java program is then called from the batch environment.

Note, the standard Java file I/O does not support reading z/OS data sets such as PDSEs, sequential data sets, or VSAM data sets. To get around this problem, the IBM JDK™ package on z/OS comes with a class library called Java Record IO (JRIO), which lets you access all z/OS data sets. For more information on JRIO, refer to *e-business Cookbook for z/OS Volume III: Java Development*, SG24-5980.

If you want to access the data that is in z/OS data sets without using JRIO:

1. Copy the input data of the batch process from the z/OS data sets to the z/OS UNIX System Services HFS files.
2. Wrap the business logic of your batch process in the Java program. It must call the APIs from Content Manager Toolkit for z/OS.
3. Copy the output files of the batch process back to the z/OS data sets.

An alternate method is to use temporary DB2 tables to store the input and output data of your batch process. You can then use JDBC™ in your Java program to read or select the input data from an input table and you can insert result rows in the output table.

### 3.7.1 Setting up data and a data model for the batch process

The data model, configuration, and data items used by the batch process can be created with the process described in the 3.5.1, “Preparing the ICM sample programs” on page 54.

We want to use just one data model for all three parts (DK APIs, CMB beans, and Batch), and use the specific data model created by the DK samples, so that we do not have to spend too much time creating data models for this chapter.

### 3.7.2 Setting up JCL for batch process

You need to include the following steps in your JCL to call the APIs from Content Manager Toolkit for z/OS:

- ▶ Copy input data
- ▶ Execute business logic

#### Copy input data

The first step is to copy the input data of the batch process from a z/OS data set to a z/OS UNIX System Services HFS file.

Example 3-7 shows you how to copy the input data.

#### *Example 3-7 Copy input data*

---

```
//CPY2HFS EXEC PGM=IKJEFT01,TIME=1439,DYNAMNBR=100,REGION=9M
//SYSPROC DD DISP=SHR,DSN=SYS1.SBPXEXEC
//SYSUT1 DD DISP=SHR,DSN=USR.INPUT.DATA
//SYSUT2 DD PATH='/tmp/input.txt',
//          PATHOPTS=(OWRONLY,OCREAT,OTRUNC),PATHMODE=SIRWXU
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
OCOPY INDD(SYSUT1) OUTDD(SYSUT2) CONVERT((BPXFX311)) FROM1047
//*
```

---

In Example 3-7, the z/OS data set USR.INPUT.DATA is copied to the HFS file input.txt in the directory tmp in z/OS UNIX System Services.

The file input.txt will be created if it does not exist, and it will be overwritten if it already exists. The data set USR.INPUT.DATA has to exist, and it must contain the input data of the batch process.

The input data of a batch process normally has many items. For example, if you need to delete a list of folders in Content Manager, the input data will contain a list of folders. Each item must be in a different record in the data set. After the copy, each item will be in a different line when viewed in an editor such as VI or OEDIT.

**Note:** A data set in z/OS may contain one or more records. An HFS file contains one continuous line; that is, it contains one stream of bits. Break characters within a file make the single line appear to be separate lines when the file is viewed.

### Execute business logic

The next step is to execute a Java program that performs the following steps:

1. Connect to the Content Manager Library Server.
2. Read the input data from the HFS file, one line at a time in a loop.
3. Execute the business logic for each line.
4. Generate output data if required.
5. Close the connection to the Content Manager Library Server.

### 3.7.3 Developing business logic in a Java program

We provide a sample Java program, presented in Example 3-8. The purpose of the program is to delete items in Content Manager.

This program has to be called from the command line with the following parameters:

- ▶ The name of the input file
- ▶ The database name of the Library Server
- ▶ The user ID defined to the Library Server
- ▶ The password of the user ID
- ▶ The item type name of the items that have to be deleted
- ▶ The name of the key attribute used to identify the items to be deleted

#### *Example 3-8 Sample Java program to delete items in Content Manager*

```
// The program will delete all items of the specified item type or entity where
// the index or search attribute is equal to any value in the input file.
//
import com.ibm.mm.beans.*;
import java.io.*;
import java.util.*;

public class Sample {

    // This program will delete all items of the specified entity or item type
    // that matches the search criteria. The search criteria is defined by the
    // attribute name and all the lines in the input file name.
    // The main method is called when the sample program is called from the
    // batch environment. It must be called with the 6 parameters listed below
```

```

public static void main(String args[]) {
    String fileName = argv[0]; // Name of file containing attribute values
    String server = argv[1]; // Database name of Library Server
    String userId = argv[2]; // User id for connection to Content Manager
    String password = argv[3]; // Password of user id
    String entity = argv[4]; // Name of item type where items must be deleted
    String attribute = argv[5]; // Name of attribute of item type to search
    BufferedReader inputFile = null;
    String inputLine = null;
    CMBCConnection connection = new CMBCConnection();
    try {
        connect(connection, server, userId, password);
        inputFile = new BufferedReader(new FileReader(fileName));
        while ((inputLine = inputFile.readLine()) != null) {
            process(connection, inputLine, entity, attribute);
        }
        System.out.println("SUCCESS:");
    } catch (Exception e) {
        System.out.println("ERROR: " + e.getClass().getName()
            + " " + e.getMessage());
    } finally {
        disconnect(connection);
        if (inputFile != null) inputFile.close();
    }
}

// This method is used to create a connection to the Content Manager
// Library Server on z/OS using the command line arguments.

public static void connect(CMBCConnection connection, String server,
    String userId, String password) throws Exception {
    connection.setDsType("ICM");
    connection.setServerName(server);
    connection.setUserid(userId);
    connection.setPassword(password);
    connection.connect();
}

// This method is used to close the connection to Content Manager

public static void disconnect(CMBCConnection connection) throws Exception {
    connection.disconnect();
}

// This method contains the main logic of the sample program. It is called
// for every line in the input file. The first step is to search for all
// items with the specified item type in the Library Server that match the
// search criteria. The search criteria matches when the attribute is
// equal to the value read from the input file. The next step is to loop

```

```

// through the search result and then to delete all the items found.

public static void process(CMBConnection connection, String inputLine,
    String entity, String attribute) throws Exception {
    CMBQueryService queryService = connection.getQueryService();
    CMBSearchResults searchResults = new CMBSearchResults();
    searchResults.setConnection(connection);
    CMBDataManagement dataManagement = connection.getDataManagement();
    search(queryService, entity, attribute, inputLine);
    searchResults.newResults(queryService.getResults());
    for (int i = 0; i < searchResults.getCount(); i++) {
        CMBItem item = searchResults.getItem(i);
        delete(dataManagement, item);
    }
}

// The search method is used to construct and execute the search query

public static void search(CMBQueryService queryService, String entity,
    String attribute, String inputLine) throws Exception {
    String condition = "[" + attribute + "=" + inputLine + "]";
    String queryString = "/" + entity + "[" + condition + "]";
    queryService.setQueryString(queryString,
        CMBBaseConstant.CMB_QS_TYPE_XPATH);
    queryService.setAsynchSearch(false);
    queryService.runQuery();
}

// The delete method is used to delete the specified item

public static void delete(CMBDataManagement dataManagement, CMBItem item)
    throws CMBException {
    dataManagement.setDataObject(item);
    dataManagement.deleteItem();
}
}

```

---

In this example, Example 3-8, the main method is executed when the class is loaded. The main method contains the following steps:

1. Create a connection to the Content Manager Library Server.
2. Open the input file and read one line at a time from it in a loop.
3. Call the process method for every line in the input file.
4. Disconnect from the Content Manager Library Server.

The *connect method* connects to the Content Manager Library Server and the *disconnect method* disconnects from it.



The *process method* may be called multiple times. It searches the Content Manager Library Server by calling the search method. The process method also loops through the search result, and then deletes every item found in it.

The *search method* is used to create the search string and execute it. The search string is constructed with the entity name, attribute, and value of the current line in the input file.

The *delete method* is called to delete an item from the Content Manager Library Server.

To keep the simplicity of the sample program, we use the minimum error checking and exception handling. When you develop your own program, make sure proper error checking and exception handling are incorporated in your code.

### 3.7.4 Calling a Java program

The next step is to execute the Java program from your batch process. Example 3-9 is sample JCL that shows you how to execute the Java program from the z/OS batch environment.

*Example 3-9 Sample JCL to execute a Java program from z/OS batch process*

---

```
//STEP2 EXEC PGM=BPXBATCH,
// PARM='SH java Sample /tmp/in.txt server userid password entity attr'
//STDIN DD DUMMY
//STDENV DD *

. /etc/profile
export JAVA_HOME=/usr/lpp/java14/J1.4
export PATH=$JAVA_HOME:$PATH
home="/u/myuid"
CLASSPATH=$home:$CLASSPATH
for i in $home/lib/*.jar; do
    CLASSPATH=$i:$CLASSPATH
done
export CLASSPATH=$home:$CLASSPATH:
/*
//STDOUT DD PATH='/tmp/&SYSUID..out',
// PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
// PATHMODE=SIRWXU
//STDERR DD PATH='/tmp/&SYSUID..err',
// PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
// PATHMODE=SIRWXU
```

---

The last step is to copy the output files back to the z/OS environment.

Example 3-10 is a sample JCL that shows you how to do it.

*Example 3-10 Sample JCL to copy output files back to z/OS environment*

---

```
//STEP3 EXEC PGM=IKJEFT01 ,DYNAMNR=300,COND=EVEN
//SYSTSPRT DD SYSOUT=*
//HFSOUT DD PATH='/tmp/&SYSUID..out'
//HFSERR DD PATH='/tmp/&SYSUID..err'
//STDOUTL DD SYSOUT=*,DCB=(RECFM=VB,LRECL=133,BLKSIZE=137)
//STDERRL DD SYSOUT=*,DCB=(RECFM=VB,LRECL=133,BLKSIZE=137)
//SYSPRINT DD SYSOUT=*
//SYSTSIN DD *
OCOPY INDD(HFSOUT) OUTDD(STDOUTL)
OCOPY INDD(HFSERR) OUTDD(STDERRL)
//
```

---

## 3.8 Using Content Manager Toolkit for z/OS from a Web environment

In this section, we present an example of a servlet you can use from a Web environment.

In this example, a request is made to the Web servlet containing the following:

- ▶ DB2 database name of the Library Server
- ▶ Content Manager user ID that must be used for the connection
- ▶ Password of the Content Manager user ID
- ▶ Entity or item type in Content Manager that must be searched
- ▶ A parameter indicating if the result must be wrapped in HTML
- ▶ Maximum number of items to be returned by the Web servlet
- ▶ Number of attributes that must be used for the search
- ▶ Searching attributes with their names, values, and operators.

The Web servlet executes the query in Content Manager and returns the result of the search in a XML structure.

You can use the HTML code in Example 3-11 to test the Web servlet.

*Example 3-11 HTML code to test sample Web servlet*

---

```
<html>
<head>
  <title>Search Item Type</title>
</head>
<body>
  <center>
    <h2>Search Item Type</h2>
    <form name="SearchItemType" method="Post"
action="http://yourserver/SearchItemType">
```

```

<table border=0>
  <tr>
    <td>Server:</td>
    <td><input name="server" type="text" value=""></td>
  </tr>
  <tr>
    <td>User Id:</td>
    <td><input name="userid" type="text" value=""></td>
  </tr>
  <tr>
    <td>Password:</td>
    <td><input name="password" type="password" value=""></td>
  </tr>
  <tr>
    <td>Entity:</td>
    <td><input name="entity" type="text" value=""></td>
  </tr>
  <tr>
    <td>Max Results:</td>
    <td><input name="max" type="text" value="50"></td>
  </tr>
  <tr>
    <td>HTML Response:</td>
    <td><input name="html" type="text" value="Y"></td>
  </tr>
  <tr>
    <td>Number of Fields:</td>
    <td><input name="fields" type="text" value="2"></td>
  </tr>
  <tr>
    <td colspan=2 align="center"><i>Field 1</i></td>
  </tr>
  <tr>
    <td>Name</td>
    <td><input name="name_1" type="text" value="LastName"></td>
  </tr>
  <tr>
    <td>Value</td>
    <td><input name="value_1" type="text" value="Smith"></td>
  </tr>
  <tr>
    <td>Operator</td>
    <td><input name="operator_1" type="text" value="="></td>
  </tr>
  <tr>
    <td colspan=2 align="center"><i>Field 2</i></td>
  </tr>
  <tr>
    <td>Name</td>

```

```

        <td><input name="name_2" type="text" value="FirstName"></td>
    </tr>
    <tr>
        <td>Value</td>
        <td><input name="value_2" type="text" value="John"></td>
    </tr>
    <tr>
        <td>Operator</td>
        <td><input name="operator_2" type="text" value="="></td>
    </tr>
</table>
<br><INPUT type="submit" value="Submit">
</form>
</center>
</body>
</html>

```

---

Figure 3-1 shows the output of the HTML file.

**Search Item Type**

Server:

User Id:

Password:

Entity:

Max Results:

HTML Response:

Number of Fields:

*Field 1*

Name:

Value:

Operator:

*Field 2*

Name:

Value:

Operator:

Figure 3-1 Output of the sample HTML file

Example 3-12 contains the source code for the Web servlet, `SearchItemType`. This servlet uses Content Manager Toolkit for z/OS to search for items with the specified search criteria. We explain each method after presenting the source code.

*Example 3-12 Source code for Web servlet, `SearchItemType`*

---

```
package za.co.ibm.servlet;

import java.io.*;
import java.lang.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
import com.ibm.mm.sdk.common.*;
import com.ibm.mm.sdk.server.*;

public class SearchItemType extends HttpServlet implements DKConstantICM {

    // This method forms the main body of the Web servlet. It is called
    // when the post request is sent from the Web page.

    public void doPost(HttpServletRequest request, HttpServletResponse response)
        throws IOException {
        DKDatastoreICM    datastore = null;
        dkResultSetCursor cursor = null;
        try {
            datastore = connect(request);
            cursor    = search(datastore, request);
            formatResult(datastore, cursor, request, response);
        } catch (Exception exc){
            displayError("Error: " + exc.getMessage(), request, response);
        } finally {
            close(datastore, cursor);
        }
    }

    // This method creates the connection to the Content Manager Library
    // Server using the connection arguments specified in the Web form.

    private DKDatastoreICM connect(HttpServletRequest request) throws DKException,
        Exception {
        DKDatastoreICM datastore = new DKDatastoreICM();
        datastore.connect(request.getParameter("server"),
            request.getParameter("userid"),
            request.getParameter("password"), "");
        return datastore;
    }
}
```

```

        // This search method builds the query string and executes the query.
        // It returns a result set or cursor that contains the search result.

private dkResultSetCursor search(DKDatastoreICM datastore,
        HttpServletRequest request) throws DKException, Exception {
    String query = "/" + request.getParameter("entity");
    if (Integer.parseInt(request.getParameter("fields") > 0) {
        query += "[" + prepare(request) + "];"
    }
    return datastore.execute(query, DKConstantICM.DK_CM_XQPE_QL_TYPE,
        init(request));
}

// This method adds the search field pairs to the query string. Each search
// field pair contains an attribute name, value, and operator.

private String prepare(HttpServletRequest request)
        throws DKException, Exception {
    String query = "";
    for (int i=1; i <= Integer.parseInt(request.getParameter("fields")); i++) {
        if (i > 1) query += " AND ";
        query += "@" + request.getParameter("name_" + i) + " " +
            request.getParameter("operator_" + i) + " " +
            request.getParameter("value_" + i) + "'";
    }
    return query;
}

// This method sets the search parameters.

private DKNVPair[] init(HttpServletRequest request)
        throws DKException, Exception {
    DKNVPair options[] = new DKNVPair[3];
    options[0] = new DKNVPair(DKConstant.DK_CM_PARM_MAX_RESULTS,
        request.getParameter("max"));
    options[1] = new DKNVPair(DKConstant.DK_CM_PARM_RETRIEVE,
        new Integer(DKConstant.DK_CM_CONTENT_ATTRONLY));
    options[2] = new DKNVPair(DKConstant.DK_CM_PARM_END, null);
    return options;
}

// This method closes the connection and the result set or cursor.

private void close(DKDatastoreICM datastore, dkResultSetCursor cursor) {
    try {
        if (cursor != null) {
            cursor.destroy();
        }
        if (datastore != null) {

```

```

        datastore.disconnect();
        datastore.destroy();
    }
} catch (Exception e) {}
}

// This method displays any exceptions that might have occurred.

void displayError(String message, HttpServletRequest request,
                  ServletResponse response) throws IOException {
    PrintWriter printWriter = response.getWriter();
    response.setContentType("text/html");
    printWriter.println("<html><body><xml id=\"search\">");
    printWriter.println("<documents error=\"1\"><message>");
    printWriter.println(message);
    printWriter.println("</message></documents></xml>\n</body></html>");
    printWriter.close();
}

// This method displays the names of the attributes of the items found in
// the search result.

private String formatHeader(DKDDO ddo) throws DKException, Exception {
    StringBuffer fields = new StringBuffer(1024);
    StringBuffer titles = new StringBuffer(1024);
    for (short i = 1; i <= ddo.dataCount(); i++) {
        if (!ddo.getDataName(i).startsWith("DK")) {
            titles.append("<th>").append(ddo.getDataName(i)).append("</th>");
            fields.append("<td><span datafld=\"field\"");
            fields.append("\" + i).append("\">></span></td>");
        }
    }
    return "<table border=1 datasrc=\"#"search\"><thead><tr> " +
        titles.toString() + "<th>DOCUMENT ID</th></tr></thead><tr> " +
        fields.toString() + "<td><span datafld=\"pid\"></span></td></tr>\" +
        "</table>";
}

// This method displays the values of the attributes of the items found in
// the search result.

private String formatItem(DKDDO ddo, HttpServletRequest request)
    throws DKException, Exception {
    StringBuffer line = new StringBuffer(1024);
    for (short i = 1; i <= ddo.dataCount(); i++) {
        Object a = ddo.getData(i);
        line.append("<field\" + i);
        if (request.getParameter("html").equalsIgnoreCase("N")) {
            line.append(" name=\"").append(ddo.getDataName(i)).append("\");

```

```

    }
    line.append(">").append(ddo.getData(i));
    line.append("</field").append("" + i).append(">");
}
return line.toString();
}

// This method is responsible for formatting the result of the search. It
//steps
// through the cursor and formats or displays each item in the result set.

private void formatResult(DKDatastoreICM datastore,
    dkResultSetCursor cursor, HttpServletRequest request,
    HttpServletResponse response) throws DKException, Exception {
    String header = "";
    StringBuffer message = new StringBuffer(1024);
    String error = request.getParameter("html").equalsIgnoreCase("Y")
        ? "" : " error=\"0\" ";
    int i = 0;
    while (cursor.isValid()) {
        DKDDO p = cursor.fetchNext();
        if (p != null) {
            String idstr = ((DKPid)p.getPidObject()).pidString();
            if (i == 0 && request.getParameter("html").equalsIgnoreCase("Y")) {
                header = formatHeader(p);
            }
            message.append("<document id=\""");
            message.append("" + (i+1)).append("\"><pid>");
            message.append(java.net.URLEncoder.encode(idstr)).append("</pid>");
            message.append(formatItem(p, request)).append("</document>");
            i++;
        }
    }
    PrintWriter printWriter = response.getWriter();
    response.setContentType("text/html");
    printWriter.print("<html><body>");
    if (i == 0) {
        printWriter.print("Info: No documents found");
    } else {
        printWriter.print("<xml id=\"search\"><documents" + error + ">" +
            message.toString() + "</documents></xml>" + header);
    }
    printWriter.print("</body></html>");
    printWriter.close();
}
}

```

---



Let us look at each method that is used in the Example 3-12.

The *doPost* method forms the main body of the Web servlet. It contains the business logic of the program, and it is called when the post request is sent from the Web page.

This method calls the connect method to create the connection to the Library Server, the search method to search the item type, the formatResults method to display the result of the search, and the close method to close the connection and cursor used by the search method.

All error reporting and exception handling is done in this method.

The *connect method* creates the connection to the Content Manager Library Server using the connection arguments specified in the Web form, shown in Example 3-11 on page 68, that was used to call the Web servlet.

The *close method* closes the search cursor and connection to the Library Server if it is open.

The *search method* uses the prepare method to build the query string that must be used for the search. After this, it will execute the query and return a cursor to the search result.

This method calls the init method to set the search parameters.

The *prepare method* is used to build the query string by concatenating the search fields in a string with the format:

```
field1 operator1 'value1' and field2 operator2 value2 ...
```

An example of this can be:

```
LastName='Smith' and FirstName='John'
```

where field 1... field x are the names of the attributes of the item type that must be searched, value 1... value x are the values the attributes, and operator 1... operator x are the relationships between the attribute fields and their values to satisfy the search criteria.

The *init method* is used to set the search parameters. The first search parameter for this example is the maximum number of items that must be returned.

The second parameter indicates that only the attributes have to be returned and not the documents.

The *formatResult method* is used to format the result of the query. It will loop through the result set returned by the search method and call the *formatItem* method for each item found.

The *formatHeader* method is called only once, just before the first item is displayed.

The *displayError method* is used to display error messages and is called whenever an exception occurs.

The *formatItem method* is used to display the attributes of an item found.

The *formatHeader method* displays the headings of the attributes of the items found.

### 3.8.1 Using WebSphere

Java programs always need to execute in a Java Runtime environment. This is also true for Web applications.

The best way to create a Java environment for a Web application in z/OS is to use IBM WebSphere Application Server for z/OS.

There are three steps involved to get a Java Web application going. First you have to develop the application and package it in a WebSphere EAR file. The third step is to configure Content Manager Toolkit for z/OS in an ASCII environment, which WebSphere requires. Refer to the Readme provided with Content Manager Toolkit for z/OS for more information.

#### Creating Websphere EAR file

The first step is to create a EAR file. You have to follow the next steps to create a EAR file using the Assembly tool of WebSphere:

1. Open the J2EE perspective to work with J2EE projects: Select **Window** → **Open Perspective** → **Other** → **J2EE**.
2. Create a new Web project: Select **File** → **New** → **Dynamic Web Project**.
3. Add the Java source code listed in Example 3-12 on page 71 to the project by creating a package in the Web project and then adding it to the package.
4. Add the HTML file listed in Example 3-11 on page 68 to the Web project.
5. Edit the web.xml file in the Web project and use the menus to add a new servlet. Pick your class name from the list of available classes and map it to the Web site specified in the HTML form.
6. Export the Web project to an EAR file by using the menus.

You can get more information on developing Web applications in the following reference:

*WebSphere Version 5 Application Development Handbook*, SG24-6993.

### **Deploying WebSphere EAR file**

The next step is to deploy the EAR file on a IBM WebSphere Application Server for z/OS. The following steps must be followed:

1. Create an enterprise server in WebSphere.
2. Create an enterprise application in WebSphere using the EAR file.
3. Start the enterprise server in WebSphere.

You can get more information on WebSphere in the reference:

*WebSphere Application Server for z/OS V5.1: Applications*, SA22-7959.



# Host integration

This chapter describes the host integration available on the z/OS platform.

We cover the following topics:

- ▶ Document routing user exit routines
- ▶ Host integration using database triggers
- ▶ Host integration using security exits
- ▶ Extend order received by Resource Manager
- ▶ Host integration using Content Manager Toolkit for z/OS
- ▶ Integration from the Windows client
- ▶ Integration from eClient
- ▶ Integration from Information Integrator for Content environment

## 4.1 Introduction

Content Manager Toolkit for z/OS can be used to accomplish host integration between Content Manager and external applications. They are normally used to extend actions or events that originate from within your external applications.

Host integration can also be accomplished to both the Library Server and Resource Manager via user exit routines. The user exits can extend events or actions that originate from within Content Manager to external applications.

The following user exits are available for host integration:

- ▶ Document routing user exit routines: Enable you to implement actions before and after work node events.
- ▶ Resource Manager user exit routines: Enable you to implement actions before and after each type of order that the Resource Manager receives.

In addition, you can achieve host integration from the Windows client, the eClient, and the Information Integrator for Content environment. This type of integration will not be discussed in detail since it is more related to the Multiplatforms environment.

## 4.2 Host integration using Content Manager Toolkit for z/OS

Use Content Manager Toolkit for z/OS whenever you need to integrate an external application to Content Manager.

In other words, you use Content Manager Toolkit for z/OS to extend an action or a event in an external application. The event or action can then query or update data in Content Manager.

You should not use Content Manager Toolkit for z/OS to extend actions or events that originate within Content Manager.

Refer to Chapter 3, “Content Manager Toolkit for z/OS” on page 45 for descriptions of available Content Manager Toolkit for z/OS components and their usage.

## 4.3 Document routing user exit routines

Document routing gives you an out-of-the box workflow solution. Although it does not provide all the functionality of a complete workflow solution, it does give you the ability to direct or route documents or folders from one work node to another.

You can construct a business process by routing documents or folders from one user or workflow step to another.

In Content Manager, a step in workflow is called a *work node*. The following is a list of available work nodes in Content Manager:

- |                             |   |
|-----------------------------|---|
| <b>Work basket</b>          | A step in the document routing process in which the document or folder waits for an action by a user or an application. |
| <b>Collection point</b>     | A step in the document routing process where the routed folder waits for the arrival of other documents.                |
| <b>Business application</b> | An external application that becomes a step in your document routing process.   |

The *Document routing user exits* give you the ability to integrate Content Manager with the external applications.

User exits, in general, give you the ability to catch or extend an event that happened within Content Manager, and you can execute your customized code before or after the event.

The following is a list of document routing events that you can extend using the user exits:

- ▶ When a document or folder enters a work node
- ▶ When a document or folder leaves a work node
- ▶ When a work node becomes overloaded
- ▶ When a document or folder is routed to a business application work node

The user exits are normally “C” programs. This is true for both the z/OS and Multiplatforms environment.

The user exit must be configured or linked to a work node with the system administration client.

### 4.3.1 When to use document routing exits

The document routing user exits must only be used when you want to extend an action or event that happens within Content Manager.

In other words, you use the document routing user exits to perform host integration from Content Manager to other external applications.

**Important:** You cannot use the document routing user exits to perform host integration from external applications to Content Manager.

You should use Content Manager Toolkit for z/OS to extend an action or event that originated from an external application to Content Manager.

The document routing exits should only be used for document routing-related host integration.

Also note that there are APIs from Content Manager Toolkit for z/OS available to integrate with the document routing processes. These APIs are normally used to control or create work items. In Content Manager, a routed document or folder is called a *work item*.

The document routing exits must only be used to catch or extend events or actions when a work item enters or leaves a work node.

There are also document routing user exits that enable you to catch or extend overload events on a work node.

One of the properties of a work node is the *overload limit* field. This is a numeric value that indicates the maximum number of work items that may wait for an action at a specific work node at any given time.

The *overload document routing user exit* is called or executed when this limit is exceeded.

Note that the overload exit is not executed when the expiration time for the work node is exceeded. Content Manager will only set a notification flag on the work item. The APIs from Content Manager Toolkit for z/OS must be used to query the value of this notification flag and initiate any actions based on its value.

Also note that the document routing user exits have to be specified for every work node where you require host integration.

At the relevant work node, you can specify one or more user exit. You have to provide the load module and the function name for each user exit.

More than one unique function can be specified in the same load module.

### 4.3.2 Sample document routing user exits

The following sample document routing user exits are provided with Content Manager.

ICMXBAU  
ICMXENT  
ICMXLEA  
ICMXOVE



You can find these examples in the PDS:

ICMV830.CMLS83.SICMSAM1

where ICMV830 is the high-level qualifier for the Content Manager installation and CMLS83 is the DB2 schema name.

The member *ICMXBAU* contains an example of a business application node user exit.

The member *ICMXENT* contains an example of a user exit that implements a work item entering a work node.

The member *ICMXLEA* contains an example of a user exit that implements a work item that leaves a work node.

The member *ICMXOVE* contains an example of a user exit that implements the overload of a work node.

All of the examples are very small and they only log each call to the exit in a file. It also logs the type of document routing exit and all the information received from Content Manager.

### 4.3.3 Development considerations

**Important:** The document routing user exits are only supported as a synchronous process. You should try to keep the processing time of the exits as short as possible.

In cases where you require the document routing user exit to perform tasks that may take a long time, you should code the user exit to spawn a new process.

Always keep in mind that the user exits are called from the Library Server.

### 4.3.4 Document routing user exit header file

The header file for the document routing user exit is in the PDS:

ICMV830.CMLS83.SICMCPY1

where ICMV830 is the high-level qualifier for the Content Manager installation and CMLS83 is the DB2 schema name.

The name of the member containing the header file is:

ICMXDRUE

The purpose of the header file is to define the structure of the data that you receive when the user exit gets control. The same structure is also used to send data back to Content Manager when the user exit returns the control to Content Manager.

Example 4-1 is the listing of the header file.

*Example 4-1 Document routing user exit header file*

---

```
typedef struct ICMCONTAINERDATA_STRUCT {
    char szContainerName[33];
    char szContainerVal[255];
} ICMCONTAINERDATA_STRUCT;

typedef struct ICMUSERSTRUCT {
    long    lUserEvent;
    char    szWPCompID[19];
    char    szWPItemID[27];
    short   sWPVersionID;
    char    szRouteSel[33];
    short   sUpdateFlag;
    short   sNumContainerData;
    struct ICMCONTAINERDATA_STRUCT **ppContainerDataStruct;
} ICMUSERSTRUCT;
```

---

The structure defined in the header file is explained in 4.3.6, “Document routing user exit interface” on page 85.

Always use the following line to include the header file in your source code:

```
#include "icmxdrue.h"
```

### 4.3.5 Document routing container data

You need to understand container data in the document routing process before you develop your own document routing user exit.

Content Manager V8.3 comes with many more built-in workflow functions. One of these workflow functions is the container data. The *container data* is the variable and value pairs that are passed to the document routing user exit each time the user exit is called. It is used to pass extra information in variables from one step in the process to the next step. Each step receives the container data and can alter or query any variables in it.

When a step is linked to a user exit, the container data is passed to the document routing user exit. The user exit can now use the values in the container data or use it later.

There is a tab called “Variables page” in the configuration of the work basket. This tab is used to define a list of variables and values that will be displayed to the user at run time.

You can define a variable as “required” with a “prompt text”. In this case, you will be prompted to enter a value in this variable during run time.

A variable can also be defined as “display to user”. In this case, the value of the variable will be displayed to the user.

Values entered during run time at one work node step can be displayed at a second work node step.

The variable and value pairs are passed to the document routing user exit each time it is called.

The document routing user exit may also alter the values of the variables before it returns to Content Manager.

### 4.3.6 Document routing user exit interface

The following is a list of the fields that are defined in a structure used by the document routing user exit:

- ▶ IUserEvent field
- ▶ szWPCompID field
- ▶ szWPItemID field
- ▶ szWPVersion field
- ▶ szRouteSel field
- ▶ sNumContainerData field
- ▶ ppContainerDataStruct field

#### **IUserEvent field**

*IUserEvent* is used to describe the type of document routing user exit. It is a long input field.

The following is a list of possible values:

- ▶ 1: When the work item has entered the work node
- ▶ 2: When the work item leaves the work node
- ▶ 3: When the work node has a overload condition

### **szWPCompID field**

*szWPCompID* is used to display the Component ID of the work item. It is a string input field with a maximum length of 33 characters.

The Component ID can be used to query the database to retrieve additional information about the work package. For example:

```
EXEC SQL SELECT ITEMID, PROCESSITEMID
        INTO :szWNItemID, :szProcessItemID
        FROM ICMUT00204001
        WHERE COMPONENTID = :szWPCompID;
```

### **szWPItemID field**

*szWPItemID* is used to display the Item ID of the document or folder item that was originally routed to this work process. It is a string input field with a maximum length of 255 characters.

### **szWPVersion field**

*szWPVersion* is used to display the version number of the work process. It is a numeric input field.

### **szRouteSel field**

*szRouteSel* is used by the document routing exit to set the route that this work package should take after control is returned to Library Server. It is a string output field with a maximum length of 33 characters.

### **sNumContainerData field**

*sNumContainerData* contains the number of container variables in the array field *ppContainerDataStruct* when the user exit is called from Library Server. It is both an input and output numeric field.

The document routing exit may alter the values of one or more of the container data variables. In this case, the variable *sNumContainerData* must contain the number of container variables altered by the user exit.

### **ppContainerDataStruct field**

*ppContainerDataStruct* contains the variable and value pairs of the container variables when the user exit is called from Library Server. It is both an input and output array field.

The document routing exit may alter the values of one or more of the container data variables. Make sure that the array contains the variable and value pairs of the container data altered by the user exit.

Note that only the variable *values* may be altered by the user exit.

### 4.3.7 Document routing user exit function prototype

The user exit must always be defined as:

```
extern long WXV2TBAUE (ICMUSERSTRUCT *pCMStruct)
```

where WXV2TBAUE is the name of the document routing user exit and you can change the name.

### 4.3.8 Document routing user exit return code

The user exit must always return a numeric long. The normal convention is to use zero when the user exit executed successfully.

A non-zero value should be returned when an error occurred.

### 4.3.9 Sample document routing user exit

Example 4-2 contains the source code for a sample document routing user exit. It can be found in:

```
ICMV830.CMLS83.SICMSAM1
```

where ICMV830 is the high-level qualifier for the Content Manager installation and CMLS83 is the DB2 schema name.

When an item leaves a work node, this user exit is called. The user exit routes the work item to a different route and it alters the container data.

*Example 4-2 Sample document routing user exit code*

---

```
# section A - Include Files
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "icmxdrue.h"

# section B- Function Prototype
extern long WXV2TLeavingUE (struct ICMUSERSTRUCT *pCMStruct) {

# section C- Displaying input
printf("Testing CM83 document routing work node leaving user exit\n");
printf("In Leaving UE, num of container data structures from LS is: %d \n",
    pCMStruct->sNumContainerData);
printf("In Leaving UE, WP Comp ID from LS is: %s \n",
    pCMStruct->szWPCompID );
```

```

printf("In Leaving UE, WP Item ID from LS is: %s \n",
      pCMStruct->szWPItemID );

for (int i = 0; i < pCMStruct->sNumContainerData; i++) {
    printf("In Leaving UE, container Data Name%d is: %s \n", i,
          pCMStruct->pContainerDataStruct[i].szContainerName );
    printf("In Leaving UE, container Data Value%d is: %s \n", i,
          pCMStruct->pContainerDataStruct[i].szContainerVal );
}

#section D - Altering container data
pCMStruct->sNumContainerData = 2
pCMStruct->ppContainerDataStruct = (ICMCONTAINERDATA_STRUCT **)
    malloc(sizeof(ICMCONTAINERDATA_STRUCT *)* pCMStruct->sNumContainerData);
strcpy(pCMStruct->pContainerDataStruct[0].szContainerName, "Variable_A");
strcpy(pCMStruct->pContainerDataStruct[0].szContainerVal, "Value_A");
strcpy(pCMStruct->pContainerDataStruct[1].szContainerName, "Variable_B");
strcpy(pCMStruct->pContainerDataStruct[1].szContainerVal, "Value_B");

#section E - Altering the route
strcpy(pCMStruct->szRouteSel, "NewRoute");

#section F - Return control to Library Server
return 0;
}

```

---

We examine the sample user exit presented in Example 4-2 in detail.

### **Section A- Include files**

This section defines the include files used by the user exit. You can add additional include files. Additional include files may be required for the Multiplatform environment.

### **Section B - Function prototype**

This section defines the name of the user exit. You may change the name of the user exit to something more appropriate for your situation.

### **Section C - Display input**

This section displays the input fields to the console. When developing your user exit, you need to implement your business logic here.

### **Section D - Alter container data**

This optional section is only required when your business logic requires you to alter any of the variable-value pairs defined in the container data.

## Section E - Alter the route

This optional section is only required when your business logic requires you to alter the route of your work item in your document routing process.

## Section F- Return control to Library Server

In this section, you end the document routing user exit with a successful return code.

### 4.3.10 How to compile and link the sample user exits

The first step is to customize the pre-link and link-edit procedure. You will find the procedure in the member ICMMLXLK in the PDS:

ICMV830.CMLS83.SICMINS1

where ICMV830 is the high-level qualifier for the Content Manager installation and CMLS83 is the DB2 schema name.

Example 4-3 is the sample listing of the procedure.

#### *Example 4-3 Sample listing of the pre-link and link-edit procedure*

```
/******  
/* ICMMLXLK - PRELINK AND LINKEDIT  
/* LOGON USER EXIT (ICMXLSLG)  
/* WORKFLOW EXIT (CM83) ON ENTER WORKNODE (ICMXENT)  
/* WORKFLOW EXIT (CM83) ON LEAVE WORKNODE (ICMXLEA)  
/* WORKFLOW EXIT (CM83) ON OVERLOAD (ICMXOVE)  
/* WORKFLOW EXIT (CM83) FOR BUSINESS APP (ICMXBAU)  
/******  
/* SETTING UP THIS CATALOGUED PROCEDURE:  
/* THE FOLLOWING SYMBOLIC PARAMETERS ARE USED IN THIS  
/* CATALOGUED PROCEDURE AND MUST BE CUSTOMIZED WHEN IT IS  
/* INSTALLED. THE MEANINGS OF THE PARAMETERS ARE AS  
/* FOLLOWS:  
/* ?ICM? -  
/* The HIGH-LEVEL QUALIFIERS OF THE DATA SETS THAT WILL  
/* CONTAIN THE OUTPUT FROM THE PRELINK AND LINK.  
/* THEY WILL BE ALLOCATED AND USED IN THIS STREAM.  
/* ?SMP? -  
/* The HIGH-LEVEL QUALIFIERS OF THE SMP/E DATA SETS  
/* WHERE LS WAS INSTALLED.  
/* ?DSN? -  
/* High-level qualifier of DB2 system libraries.  
/* ?OBJLIB? -  
/* THE FULLY QUALIFIED DSNAM FOR THE OBJECT FROM THE  
/* COMPILE OF THE SOURCE FOR ICMXLSLG OR OTHER MODULES  
/* ?LOADLIB? -
```

```

/* FULLY-QUALIFIED LOAD LIBRARY DATA SET
/* THIS IS THE LOAD LIBRARY WHERE THE LMOD FOR
/* ICMXLSLG OR OTHER MODULES WILL BE LINKED.
/* NEEDS TO BE ALLOCATED
/* Note: Do not write to the SMP/E libraries
/*****
/* &HLQ1 The HIGH-LEVEL QUALIFIERS OF THE DATA SETS THAT
/* WILL CONTAIN THE OUTPUT FROM THE PRELINK
/* AND LINK.
/* THEY WILL BE ALLOCATED AND USED IN THIS STREAM.
/*
/* &HLQ2 - HIGH-LEVEL QUALIFIER OF THE SMP/E DATA SETS
/* WHERE Library Server WAS INSTALLED
/* &LOADLIB - FULLY-QUALIFIED LOAD LIBRARY DATA SET
/* THE SYMBOLIC PARAMETERS DIRECTLY FOLLOW THIS COMMENT
/* BLOCK AND MOST LIKELY REQUIRE CUSTOMIZATION BY YOUR
/* INSTALLATION. *
/*****
//ICMMLXLK PROC HLQ1='?ICM?',
// HLQ2='?SMP?',
// OBJLIB='?OBJLIB?',
// LOADLIB='?LOADLIB?',
// LPARM='AC=1,RENT,AMODE=31,RMODE=ANY',
// LPARM2='MAP,LET,LIST,NCAL,RENT',
/*****
// LIBPFX='CEE.',
// LIBRUN='SCEERUN',
// EDCMSGSGS='SCEEMSGP',
// LANG='EDCPMSG',
// LIBBASE='SCEELKD',
/*****
// DSNHDSR='?DSN?.',
// DSNLOAD='SDSNLOAD',
// WRKSPC='(32000,(30,30))'
/*****
/* PRE-LINKEDIT
/*****
//S1 EXEC PGM=EDCPRLK,REGION=40M,PARM='MEMORY,MAP'
//STEPLIB DD DSN=&LIBPFX&LIBRUN,DISP=SHR
//SYSMSGSGS DD DSN=&LIBPFX&EDCMSGSGS(&LANG),DISP=SHR
//SYSLIB DD DSN=&LIBPFX.SCEECPP,DISP=SHR
//OBJLIB DD DISP=SHR,DSN=&OBJLIB
//SYSMOD DD DSN=&HLQ1..PRLOUT(&MEMBER),DISP=(NEW,CATLG),
// UNIT=SYSDA,SPACE=(TRK,(3,2,2)),
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120)
//SYSDEFSD DD DSN=&&DEFSD(&MEMBER),DISP=(NEW,PASS),
// UNIT=SYSDA,SPACE=(TRK,(3,2,2)),
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120)
//SYSOUT DD SYSOUT=*

```



```

//SYSPRINT DD SYSOUT=*
//SYSIN    DD DSN=&HLQ2..SICMINS1(&PLKCTL),DISP=SHR
//*
/*****
/* NCAL LINK
/*****
//S2 EXEC PGM=IEWL,PARM='&LPARM2',
//      REGION=20M,
//      COND=(5,LE,S1)
//SYSLIB DD DISP=SHR,DSN=&DSNHDRS&DSNLOAD
//      DD DSN=&LIBPFX&LIBBASE,DISP=SHR
//SYSLIN DD DSN=&HLQ1..PRL0UT(&MEMBER),DISP=SHR
//SYSLMOD DD DSN=&HLQ1..ICMOBN1(&MEMBER),DISP=(NEW,CATLG),
//          UNIT=SYSDA,SPACE=(TRK,(3,2,2)),
//          DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120)
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD SPACE=&WRKSPC,UNIT=SYSDA
//SYSIN DD DUMMY
//*
/*****
/* LINKEDIT
/*****
//S3 EXEC PGM=IEWL,PARM=('LIST,MAP,DYNAM=DLL',
//      '&LPARM'),REGION=20M,
//      COND=(5,LE,S2)
//SYSLIB DD DISP=SHR,DSN=&DSNHDRS&DSNLOAD
//      DD DSN=&LIBPFX&LIBBASE,DISP=SHR
//      DD DSN=&HLQ1..ICMOBN1,DISP=SHR
//      DD DSN=&HLQ2..SICMLOD1,DISP=SHR
//SYSDEFSD DD DSN=&&DEFSD(&MEMBER),DISP=(OLD,PASS)
//SYSLIN DD DDNAME=SYSIN
//SYSLMOD DD DISP=SHR,DSN=&LOADLIB(&MEMBER)
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD SPACE=&WRKSPC,UNIT=SYSDA
//SYSIN DD DSN=&HLQ2..SICMINS1(&LINKCTL),DISP=SHR
//*
//S99 EXEC PGM=IEFBR14,
//      COND=(4,LE,S3)
//DD1 DD DSN=&HLQ1..PRL0UT,DISP=(OLD,DELETE)
//DD2 DD DSN=&HLQ1..ICMOBN1,DISP=(OLD,DELETE)

```

You need to change the following parameters by executing a search and replace command in TSO:

► ?ICM?

This value indicates the high-level qualifier for the data set that contains the output of the pre-link and link-edit procedure.

You do not have to allocate these data sets. It is created by the procedure.

► ?SMP?

This value indicates the high-level qualifier for the SMP/E data sets where Content Manager Library Server is installed.

► ?DSN?

This value indicates the high-level qualifier for the DB2 system libraries.

► ?OBJLIB?

This value indicates the fully qualified data set name that contains the source code of your document routing user exit.

Please note that this data set is a PDS and that each member contains the source code of a different user exit. The name of this member is also used as the member name of the load library for your user exit.

► ?LOADLIB?

This value indicates the fully qualified data set name that will be used as the load library for your user exit. Your source code is compiled and linked to this load library.

This data set has to be allocated by you before you execute this step.

Do not use the SMP/E data sets that contain the Library Server installation for this step.

The next step is to run the pre-link and link step. The JCL for this step can be found in the member ICMMLXLJ in the PDS:

ICMV830.CMLS83.SICMINS1

where ICMV830 is the high-level qualifier for the Content Manager installation and CMLS83 is the DB2 schema name.

Example 4-4 is the JCL example.

*Example 4-4 JCL example for pre-link and link step*

---

```
//ICMMLXLJ JOB (XXXX),'LINK EXIT',
//          MSGLEVEL=(1,1),MSGCLASS=?OUTC?
//*****
//* JOB TO LINK LOGON EXIT PROGRAM, ICMXLSLG
//*****
//* CUSTOMIZE THE FOLLOWING FIELDS:
//*   ?ICM?
//*           - HIGH-LEVEL QUALIFIER OF CM8 LIBRARIES
//*   ?OUTC?
//*           - JOB OUTPUT MESSAGE CLASS
//*****
```

```
//LIB JCLLIB ORDER='?ICM?.SICMINS1'  
//ICMMHLLL EXEC PROC=ICMMLXLK, MEMBER=ICMXLEA, PLKCTL=ICMMHPLL,  
//          LINKCTL=ICMMHLLL
```

---

You need to change the following parameters in the JCL before you submit the job by executing a search and replace command in TSO:

► ?ICM?

This value indicates the high-level qualifier for the data sets that contain the libraries for Content Manager Library Server.

► ?OUTC?

This is the message class that is used for the output of your job when you submit the JCL.

### 4.3.11 Configure document routing user exit

You need to define the document routing user exit to Content Manager Library Server before you can use it.

As mentioned earlier, the document routing user exit has to be associated with a work node. There are three different types of work nodes:

- Work basket
- Collection point
- Business application

The configuration and the way the user exit works do not differ for the different types of work nodes.

The following two values have to be specified when you configure the user exit:

- The name of the load library that contains the linked user exit
- The name of the function in the user exit that has to be called from the Library Server

The system administration client that runs on the Windows platform is used to configure the document routing user exits.

#### Configuring the user exit in a work basket

Figure 4-1 shows an example of the document routing user exit defined in a work basket node. The user exit is called when an item leaves the work basket.

	Link library name	Function name
Overload:		
Entering:		
Leaving:	ICMV830.LOADLIB(ICMMLXLK)	WXV2LeavingUE

*Figure 4-1 Definition of document routing user exit in a work basket*

The following is a list of values used in the example:

- ▶ **WB3**  
This is the name of the document routing work basket.
- ▶ **ICMV830.LOADLIB(ICMMLXLK)**  
This is the name of the load library containing the executable document routing user exit.
- ▶ **WXV2LeavingUE**  
This is the name of the function that is executed in the document routing user exit.

### **Configuring the user exit in a collection point**

Figure 4-2 shows an example of the document routing user exit defined in a collection point work node. The user exit is called when an item leaves the collection point.

	Link library name	Function name
Overload:		
Entering:		
Leaving:	ICMV830.LOADLIB(ICMMLXLK)	WXV2LeavingUE

Figure 4-2 Definition of document routing user exit in a collection point

The following is a list of values used in the example:

- ▶ ICMV830.LOADLIB(ICMMLXLK)  
This is the name of the load library containing the executable document routing user exit.
- ▶ WXV2LeavingUE  
This is the name of the function that is executed in the document routing user exit.

### Configuring the user exit in a business application

Figure 4-3 shows an example of the document routing user exit defined in a business application work node.

**New Business Application**

Name: \* BUSSNODE

Description: Sample Business Node

Long description: Sample Business  
Node Long  
Description

Access control list (ACL): \* PublicReadACL

Entry point:

Link library name: \* ICMV830.LOADLIB(ICMMLX)

Function name: \* WXV2LeavingUE

OK Cancel Apply Help

Figure 4-3 Definition of a document routing user exit in a business application

The following is a list of values used in the example:

- ▶ BUSSNODE  
This is the name of the business application work node.
- ▶ ICMV830.LOADLIB(ICMMLX)  
This is the name of the load library containing the executable document routing user exit.
- ▶ WXV2LeavingUE  
This is the name of the function that is executed in the document routing user exit.

### 4.3.12 Using document routing user exits

You need to define a data model and document routing process before you can use the user exit.

Chapter 3, “Content Manager Toolkit for z/OS” on page 45 contains the steps on how to define this using the Java sample programs, including how to set up your development environment for running the sample Java programs.

## 4.4 Host integration using database triggers

Host integration to an external application is normally done with database triggers when there are no user exits or APIs available, and when you also do not have access to the source code of the external application.

5.7, “Additional options with DB2 triggers” on page 138 presents more information on how to use database triggers in Content Manager for z/OS.

**Important:** You need to be very careful using database triggers.

The following are potential problems:

- ▶ Performance problems can occur when you add additional overhead.
- ▶ Timeout problems may be introduced to your environment.
- ▶ New versions of Content Manager may have different database structures.

*Use database triggers at your own risk.* It is not the responsibility of the IBM DB2 Content Manager Support Team or the ITSO Redbook Team to support any issues from this type of implementation. We only mention the database trigger here to inform you of the various types of options available to achieve host integration.

## 4.5 Host integration using security exits

Host integration can also be done using security exits. Host integration using security exits is very handy for logging audit information because it is called for every action that the user initiates from the client application.

Chapter 5, “Security and exits” on page 111 provides more information on how to use the security exits of Content Manager for z/OS.

You need to be very cautious when using these exits because they are called every time a user performs an action. Your system can become very slow if you add too much overhead in the security user exits.

## 4.6 Extend order received by Resource Manager

The Content Manager Resource Manager provides user exit routines to catch events or actions. The events or actions are normally related to orders or requests received by Resource Manager.

You may use the user exits to catch or extend orders received by Resource Manager. Resource Manager provides two types of exits:

- ▶ Before an action or request to Resource Manager
- ▶ After an action or request to Resource Manager

The names of Resource Manager user exits are reserved. You may *not* use your own names for the functions when you develop user exits.

The following orders to Resource Manager may be extended by an user exit:

- ▶ When a new object is stored in Resource Manager
- ▶ When an object is retrieved from Resource Manager
- ▶ When an object in Resource Manager is replaced
- ▶ When the storage information of an object is changed
- ▶ When the storage information of an object is queried
- ▶ When an object is deleted from Resource Manager
- ▶ When a pre-fetch of an object occurs

### 4.6.1 When to use Resource Manager user exits

There are several reasons why you may want to use Resource Manager user exits.

One reason is to implement an audit log. This is normally done to log any request by a user to an object in Resource Manager.

Another reason is to extend the storage subsystem. This is done when you want to make a copy of an object to an external application or system, or to extend the pre-fetch rules of your storage subsystem.

You may also want to change a collection name when storing an item due to the fact that a collection is full. To get more detailed information on this topic, see 4.6.7, “Dealing with large Resource Manager collections” on page 104.

Resource Manager is implemented using either OAM or TSM. Both these products are very rich in functionality. Although Resource Manager user exits provide extra functions, we recommend using the functions that come with OAM and TSM to manage your resource objects whenever possible.

### 4.6.2 Predefined user exit names

The following is a list of predefined user exit names that have to be used when you develop your own Resource Manager user exits:



► PRESTOR

This user exit is used to extend the store of an object in Resource Manager. It is always called before the store of a new object.

► POSTSTOR

This user exit is used to extend the store of an object in Resource Manager. It is always called after the store of a new object.

► PRERTRV

This user exit is used to extend the retrieval of an object from Resource Manager. It is always called before the retrieval of an object.

► POSTRTRV

This user exit is used to extend the retrieval of an object from Resource Manager. It is always called after the retrieval of an object.

► PREREPL

This user exit is used to extend the replacement of an object in Resource Manager. It is always called before the retrieval of an object.

► POSTREPL

This user exit is used to extend the replacement of an object in Resource Manager. It is always called after the retrieval of an object.

► PRECHSM

This user exit is used to extend the changing of storage management system information. It is always called before the information is changed.

► POSTCHSM

This user exit is used to extend the changing of storage management system information. It is always called after the information is changed.

► PREQUSM

This user exit is used to extend the query of storage management system information of an object. It is always called before the information is queried.

► PREDELT

This user exit is used to extend the deletion of an object in Resource Manager. It is always called before the deletion of an object.

► POSTDELT

This user exit is used to extend the deletion of an object in Resource Manager. It is always called after the deletion of an object.

- ▶ PREPFCH

This user exit is used to extend the pre-fetch of an object in Resource Manager. It is always called before the pre-fetch of an object.

- ▶ POSTPFCH

This user exit is used to extend the pre-fetch of an object in Resource Manager. It is always called after the pre-fetch of an object.

### 4.6.3 Resource Manager user exit interface

The Resource Manager user exit is called with one parameter that contains the following:

- ▶ The information that is passed to the HTTP server when the request is made to Resource Manager.
- ▶ The information that Resource Manager used to access the physical object in the storage management system. This can be either TSM or OAM.
- ▶ The information that is returned from the storage management system after the access of the physical object.

You may change the collection and pre-fetch collections during the execution of your user exit. The rest of the values should be read-only.

Your user exit should return zero when it is successful. In case of an error, it should return a non-zero value.

### 4.6.4 Resource Manager user exit function prototype

The following is an example of a function prototype for your Resource Manager user exit:

```
int PRESTOR (void *pvHTTPHeader) {
```

In this example, PRESTOR is the name of the user exit.

### 4.6.5 Example of Resource Manager user exit

Example 4-5 shows an example of the Resource Manager user exit that extends the store of an object. The user exit is called before an object is stored in Resource Manager. Within the user exit, the collection name is changed under certain conditions.

The source code for this sample Resource Manager user exit can be found in:

ICMV830.CMLS83.SICMSAM1

where ICMV830 is the high-level qualifier for the Content Manager installation and CMLS83 is the DB2 schema name.

*Example 4-5 Sample Resource Manager user exit: Extend object store*

---

```
// Section A - Define pragma
#pragma linkage (PRESTOR, fetchable)

// Section B - Include files
#include <stdio.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// Section C - Function prototype
int PRESTOR (void *pvHTTPHeader) {

// Section D - Structure for Resource Manager list
typedef struct {
    char szRMName[30]; //Resource Manager Name
    char szProtocol[6]; //Protocol Type (for example, http)
    char szHostName[256]; //Host name where RM resides
    char szPort[5]; //Port number where RM listens
    char szPath[256]; //Path to RM's executable
    char szCollName[44]; //Target collection where object is replicated
} RCLIST;

// Section E - Structure for input of user exit
typedef struct {
    char pszObjID[135];
    char pszTranID[135];
    char pszUserID[135];
    char pszAction[117];
    char pszObjToken[135];
    char pszItemID[81];
    char pszCollName[135];
    char pszPrefetchCollName[135];
    char pszVersionID[12];
    char pszObjLen[117];
    char *pszObj;
    char pszObjOffset[117];
    long lObjLen;
    char pszMgmtClass[27];
    char pszStorageClass[27];
    char pszRetentionPrd[15];
    char pszNewCollName[135];
    char pszNewMgmtClass[27];
    char pszNewStorageClass[27];
    char pszNewRetentionPrd[15];
```

```

char pszPartNum[12];
char pszDB2SubsysID[15];
char pszDB2PlanName[27];
void *pvTokenArea;
short sTrace;
unsigned char *handle;
char pszCreateDate[81];
char pszCreateTime[81];
char pszLastReferencedDate[81];
char pszExpirationDate[81];
char pszEstimRetrievalTime[81];
char pszLibName[126];
char pszTempObjID[135];
short sCommitRoll;
char pszTrkTbITS[81];
short sNumReplicas;
RCLIST rcList[30];
char pszIPAddress[93];
char pszTokenKey[300];
int sTokenDuration;
short sPreFuncExitPgm;
short sPostFuncExitPgm;
short sObjTokens;
char szUpdateDate[81];
char pszUpdateTime[81];
short sPrefetchEnabled;
char szTempCollName[ICM_RM_COLLNAME_LENGTH_WITHNULL];
} HTTP_HEADER_DATA, *P_HTTP_HEADER_DATA;

// Section E - Main body
printf("Trace message...entering pre-store exit program...\n");
pHTTPHeaderData = (HTTP_HEADER_DATA*)pvHTTPHeader;
//If the incoming collection name is CLLCT001, override with CLLCT003.
if(strcmp (pHTTPHeaderData->szCollName, "CLLCT001") == 0)
    strcpy (pHTTPHeaderData->szCollName, "CLLCT003");
return (0); //Return good return code.
}

```

---

We examine the code presented in Example 4-5 in detail.

## Section A - Define pragma

This section instructs the “C” compiler to link the user exit as a callable module.

## Section B - Include files

This section contains standard include files.

## Section C - Structure for the Resource Manager list

This section is for future use and should not be used.

## Section D - Structure for the input of the user exit

This section contains the data that is passed from Resource Manager to the user exit. The same structure is used for all the user exits of Resource Manager.

## Section E - Function prototype

This section contains the name of your user exit. Note that you may only use the predefined list of names.

## Section F - Main body

This section contains an example on how to log information received by Resource Manager.

The example also shows you how to change the collection where the object or document is stored in the storage management system.

In the end, the example returns a value of zero. This means that it is successful, and Resource Manager will store the object using the modified collection name.

### 4.6.6 How to use Resource Manager user exits

You may use the following steps to use your Resource Manager user exits:

1. Allocate a load library and link the user exit in it.

Allocate a load library, for example: IBM.USEREXIT.LOADLIB. Use the C compiler on z/OS to compile and link the example to your load library.

Note, the exit program can be written in any language using standard linkage: C/C++, PL/1, Assembler, or COBOL, but not Java.

2. Configure the HTTP server to use the load library.

You need to alter the load procedure for your IBM HTTP server to load the library containing the user exit.

The following is an example that shows you how to modify the load procedure:

```
//STEPLIB DD DSN=DB2C.DSNLOAD,DISP=SHR  
// DD DSN=IBM.USEREXIT.LOADLIB,DISP=SHR
```

3. Configure Resource Manager to use the user exit.

You need to do an SQL update to the Resource Manager database to configure the user exit.

The following is an example SQL that shows you how to enable the PRESTOR Resource Manager user exit:

```
db2 update SCHEMA NAME.icrmcontrol set PRESTOREXITPGM = 1
```

4. Use Resource Manager user exit.

You need to stop and start both Resource Manager and the IBM HTTP server to activate the user exit.

The user exit is called every time an object is stored in Resource Manager.

## 4.6.7 Dealing with large Resource Manager collections

In a very large implementation of Content Manager for z/OS, you may run into a problem with collections that get full.

You can have one or more collections in an OAM storage group (database). Sometimes your system may fill up a storage group faster than you have anticipated based on the initial design.

Having multiple large collections within the same OAM storage group as well as having multiple item types assigned to the same collection can accelerate the growth of an OAM storage group. Content Manager and OAM do not provide any proactive feedback to inform you that an OAM storage group (database) is near capacity.

It is best to have large collections in their own storage group and item types assigned to their own collection.

DB2 tables exist in table spaces. DB2-segmented table spaces, which have a 64 GB limit, provide the best performance, but not the highest capacity.

For example:

- ▶ A single collection in a single OAM storage group, using segmented table spaces, would have a limit of 64 GB.
- ▶ Multiple collections in a single OAM storage group, using segmented table spaces, would have a limit of 64 GB.

Partitioned table spaces provide a higher capacity (greater than 16 TB for DB2 V7, and greater than 128 TB for DB2 V8), but do not perform as well as segmented tables.

Note, if your Resource Manager is configured to use TSM, you should not have this problem.

From a system management point of view, regardless of your system setup, we recommend that you do not work with collections that are bigger than 64 GB, because data backup and restoration can get difficult to perform.

Resolving a group that is full is an administration issue. In OAM, create a new group and define it as a new collection in your Resource Manager. You can then start using it in Content Manager.

Before we continue, let us review how Content Manager uses default collections. In a system administration client, you specify if you want to link the collections to users or to the item types.

If you have remote users with their own Resource Manager close to them, you normally would link the collections to the users.

If you only have one Resource Manager and have multiple departments sharing the same Resource Manager, it is better to link the collections to the item types.

Remember that the default collection is only used when new resource items are created. When you specify a new default collection for an item type, the documents already stored in the previous collection will stay in that collection and you will still be able to view it from the clients. All new documents will now be stored in the new collection in the Resource Manager.

In a very large implementation of Content Manager for z/OS, you may have many item types.

One method to switch collections when a group gets full is to use the system administration client and to manually change the collection for each user or for an item type.

A second method is *to use the Resource Manager user exit*. In Example 4-5 on page 101, in the sample Resource Manager user exit that extends object store, we show you how to *overwrite* the collection name that will be used to store the document. You may combine this example with a lookup table to read the current collection name.

In the case that you switch the collection for an item type, you only need to modify the collection name in the lookup table.

A third method is to use Content Manager Toolkit for z/OS to change the collection name. The advantage of this method is that you only have to run the program once each time you switch groups. It is always best to use strict naming conventions in a large implementation.

For example, you can use a prefix in the name of the item types to group together the item types of a department. Refer to Appendix C, “Sample code to change

default collections” on page 375 for the source code of an example that shows you how to change the default collection names of all the item types starting with a specified prefix in the name.

## 4.7 Integration from the Windows client

This section is not related to the z/OS environment. We, therefore, will not go into too much detail. To get more information on this topic, refer to the following reference:

- *IBM DB2 Content Manager Enterprise Edition: Client for Windows Programming Reference, SC27-1337*

You can change the behavior of the Windows client by developing user exit routines. These exits do not execute on z/OS and there is no equivalent for them on z/OS.

The user exits for the Windows client are always dynamic-link libraries called DLL files. The DLLs are normally developed using C programming language.

DLL distribution is the biggest problem with the client-side user exits. It is best to develop the Windows client user exits using a very generic structure. The client user exit should only be an interface to a centralized system, for example, DB2 on z/OS.

The interface to the centralized system can be done on a data or transactional level. It is not possible to define a single method of integration between the client user exit and a centralized system. It is best to use your standard method of integration.

The only thing that you should avoid is storing data hard-coded in the Windows client user exit. This causes distribution problems each time the data changes. It is also very difficult to manage different versions of the Windows client user exit on multiple client workstations.

### 4.7.1 A practical example

The easiest way to explain this is by using a practical example of a drop-down list.

For example, you have a city attribute in the client document item type. You do not want your users to manually type the name of the city. It is much easier to select the name of the city from a drop-down list.



This functionality is implemented using a Windows client user exit with one of the following methods:

- ▶ You can hard code the names of the cities in the user exit DLL file.
- ▶ You can code the user exit DLL file to read the cities from a properties file stored on a centralized system.
- ▶ You can code the user exit DLL file to select the cities from a database stored on a centralized system using a database interface such as ODBC.
- ▶ You can code the user exit DLL file to get the cities from a external application on a centralized system using a transaction interface such as Web services.

## 4.7.2 Available Windows client user exits

The following is a list of Windows client user exits and when to use them:

- ▶ Add items to action menu

This user exit is used to add extra items in the action menu of the Content Manager client for Windows. An external application is called when the user selects the added menu items.

- ▶ Alternate search

This user exit is used to replace the normal search function in the Content Manager client for Windows. The user exit has to return a collection or a search result and the Windows client will then display it.

- ▶ Change SMS

This user exit is called when the item type of an item is changed in the Content Manager client for Windows.

- ▶ Enable or disable menu items

This user exit is called each time the menu items in a menu are displayed by the Content Manager client for Windows. The user exit must decide if the menu item is enabled or disabled for the current user.

- ▶ Get attribute value list

User may use this exit to extend an attribute to display a drop-down or combo list. It extends the attribute on the search, store, and new folder screens. The exit is used to force or encourage the user to select a value from a predefined list.

- ▶ Process action

This user exit is called when a user-defined action is selected from the document routing process menu.

- ▶ Query sort

This user exit is used to replace the sort order for items in a work list or folder in the Content Manager client for Windows.

- ▶ Save record

This user exit is called when the user saves changes to the attributes of a document or a folder in the Content Manager client for Windows.

- ▶ User option function

This user exit is called when the user selects a menu item added with the “add items to action menu” user exit.

## 4.8 Integration from eClient

This section is also not related to the z/OS environment. To get more information on this topic, you need to refer to the following reference:

- ▶ *IBM DB2 Content Manager Enterprise Edition V8.3: Installing, Configuring and Managing the eClient*, SC27-1350

The eClient provides you with a Web interface to Content Manager for z/OS.

One type of integration on eClient can be for enhancing the look and feel. The integration can be done by changing the Java servlet pages (JSP). This integration by far is the simplest. The biggest problem with this type of implementation is that you might overuse it.

You should avoid changing the JSP files of eClient to implement integration that can be done using user exits or Content Manager Toolkit for z/OS. Although it may save you time in the short term, it can become a problem when you upgrade to the next version of eClient.

The following integration can be accomplished by changing the JSP files of eClient:

- ▶ Changing the look and feel of eClient by adding your own graphics, colors, and logos.
- ▶ Web equivalent coding for any user exits that you might have implemented for the Content Manager client for Windows.

## 4.9 Integration from Information Integrator for Content environment

The Content Manager Information Integrator for Content (II for Content) environment is only available for Multiplatforms and is not available on z/OS. There is a connector available in Information Integrator for Content that can be used to connect to Content Manager for z/OS.

Information Integrator for Content can be used to create a federated view of multiple document repositories including Content Manager for z/OS.

For more information on Information Integrator for Content, refer to the following publications:

- ▶ *IBM DB2 Content Manager Enterprise Edition V8.3: Application Programming Guide*, SC27-1347
- ▶ *IBM DB2 Content Manager Enterprise Edition V8.3: Application Programming Reference*, SC27-1348

Integration from the Information Integrator for Content environment can be complicated because it runs on a different platform. You should use a middleware product such as IBM MQSeries in case you do attempt to use it.

Information Integrator for Content has a rich set of APIs available to implement almost any action in Content Manager for z/OS.

The APIs in Information Integrator for Content can be called from both a Java and C++ environment. It also supports the Microsoft .NET interface.



## Security and exits

In this chapter, we cover special topics related to Content Manager security and exits. This chapter should be used in conjunction with the information provided in:

- ▶ *IBM DB2 Content Manager for z/OS V8.3: System Administration Guide*, SC27-1335

In this chapter, we cover the following topics:

- ▶ Security in Content Manager overview
- ▶ RACF logon exit implementation: Replace existing password validations
- ▶ ACL user exit routines overview
- ▶ ICMACLPriExit: Replace existing ACLs
- ▶ ICMGenPriExit: Replace user's general privileges
- ▶ Writing exits in non-C programming languages
- ▶ Additional options with DB2 triggers for security and key value validations
- ▶ Resource Manager exits

## 5.1 Security in Content Manager overview

Content Manager for z/OS enables you to modify the standard security model through the use of exits.

The exit programs listed in Table 5-1 are available for you to use and adapt to your specific needs with regards to security.

Table 5-1 List of security exit programs

Exit program	Description
ICMXLSLG ICMMRACF	Replace Content Manager validation of passwords by validation through RACF or similar security software.
ICMACLXT	Replace Content Manager use of ACLs with your own security checking.
ICMGEXT	General security exit to validate a user's general privilege.

For information on the standard Content Manager security model including ACLs, user authorization, and authentication, refer to:

- *IBM DB2 Content Manager for z/OS V8.3: System Administration Guide*, SC27-1335

## 5.2 RACF logon exit implementation

When you log on to a system, the DB2 stored procedure ICMLogon is called, which executes the program ICMPLSLG. The definition of this stored procedure is in the installation job ICMMLSCR, which is shown in the extract from the job in Example 5-1.

Example 5-1 DB2 stored procedure ICMLogon

```
create procedure ICMLS.ICMLOGON
(
  Out      1RC              INTEGER,
  Out      1Reason          INTEGER,
  Out      1ExtRC           INTEGER,
  Out      1ExtReason       INTEGER,
  In       sTraceLevel      SMALLINT,
  In       1Reserved1       INTEGER,
  InOut    szUserInfo       VARCHAR(254),
  InOut    szUserToken      CHAR(32),
  InOut    1Reserved       INTEGER,
  InOut    szLanguageCode   CHAR(3),
  InOut    szUserID         VARCHAR(32),
```

```

In      szPassword      CHAR(48) FOR BIT DATA,
In      szNewPassword   CHAR(48) FOR BIT DATA,
In      szApplication   VARCHAR(32670),
Out     szUserName      VARCHAR(128),
Out     lUserPrivSetCode INTEGER,
Out     lGrantPrivSetCode INTEGER,
Out     lDfltItemACLCode INTEGER,
Out     sDfltRMCode     SMALLINT,
Out     sDfltSMSCollection SMALLINT,
Out     szUserPrivSetName VARCHAR(32),
Out     szGrantPrivSetName VARCHAR(32),
Out     szDfltACLName   VARCHAR(32),
Out     szDfltRMName    VARCHAR(128),
Out     szDfltSMSCollection VARCHAR(44),
Out     sSystemFlag     SMALLINT,
Out     lMaxResultSetSize INTEGER,
Out     szPlatform      SMALLINT,
Out     lUserDomainID   INTEGER
)
COLLID ICMPC1
WLM ENVIRONMENT WLMICM1
DYNAMIC RESULT SETS 2
LANGUAGE C PARAMETER STYLE DB2SQL
NO DBINFO FENCED STAY RESIDENT YES PROGRAM TYPE SUB
RUN OPTIONS 'STAC(,,ANY,,)ALL31(ON)'
EXTERNAL NAME ICMPLSLG;

```

---

When the Content Manager Library Server program, ICMPLSLG, executes at logon time, it calls the exit program, ICMXLSLG. If this program is not available, the password validation continues based on the password registered in the Content Manager Library Server table, ICMSTUSERS, where the password is saved in an encrypted format.

If you wish to use RACF for password validation, you must make the exit program available in a STEPLIB of the Workload Manager (WLM) where the logon program executes. The WLM is specified by the **WLM ENVIRONMENT** part of the stored procedure definition (which is bolded in Example 5-1). You tailor this value during the installation process in installation job ICMLLSCR.

The RACF support is implemented via two programs shipped in the xxx.SICMSAM1 library of the Content Manager install data sets:

- ▶ **ICMXLSLG**: This is a C program called by the Content Manager logon program. It receives the parameter list passed by Content Manager which contains the following information:
  - Language code
  - User ID

- Decrypted password
- New decrypted password (if new password is provided)
- Application name
- User domain
- Path in the LDAP server for this user
- Library Server database name
- Database schema name of the Library Server

If there is no C programming expertise in your company, you do not have to use the sample C program as is. You can implement ICMXLSLG in COBOL or other Language Environment® (LE) supported languages.

For information, refer to 5.6, “Writing exits in non-C programming languages” on page 132.

- ▶ ICMMRACF: This assembler program is called by the ICMXLSLG in order to perform the actual RACF validation of user ID and password. Validation is done through a call to the RACF function RACROUTE.

### 5.2.1 Activating RACF exit

In order to activate the RACF exit, you must first compile and link-edit the two programs, ICMXLSLG and ICMMRACF. You then need to make them available in a STEPLIB allocated to the workload manager (WLM) used by the Library Server.

The JCL used to perform the compilation and linkage of the programs can be found in the xxx.SICMSAM1 library. For a complete list of the jobs you must run, refer to the manual:

- ▶ *IBM DB2 Content Manager for z/OS: Planning and Installing Your Content Management System*, GC18-7698

### 5.2.2 APF authorizing Content Manager load library

If you use the RACF exit, the Content Manager load library used in the STEPLIB list of the WLM must be *APF-authorized*. If you have multiple libraries in your STEPLIB concatenation, they must *all* be APF-authorized.

### 5.2.3 Using security managers other than RACF

If you need to perform user validation by means other than RACF, you can do this by either:

- ▶ Changing the program, ICMMRACF, to call another security manager



- ▶ Changing the program, ICMXLSLG, to either:
  - Call a program other than ICMMRACF (make sure that this program is made available in the WLM STEPLIB).
  - Include the security validation directly in the ICMXLSLG program.

## 5.2.4 Users not validated by the security exit

The following types of users are not validated via the security exit since they perform a direct connect to the DB2 database:

- ▶ Super administrator users
- ▶ Connect users (such as the ICMCONCT user)

**Important:** Super administrator and connect users are not validated against the security exit. This is important if you plan to use the logon exit for additional functions such as recording logon statistics.

## 5.2.5 Importing user definitions from RACF

If you have users already defined in RACF, you can import user definitions from RACF and set up your Content Manager system to use these definitions.

There is a set of sample jobs in the xxx.SICMINS1 library that demonstrates how to accomplish this. These jobs are listed and described in Table 5-2.

*Table 5-2 Jobs used to import RACF-defined users to Content Manager*

Member in SICMINS1	Purpose
ICMMBKUP ICMMDATA ICMMSORT	Extracts user information from RACF and sorts it: <ol style="list-style-type: none"> <li>1. Job ICMMBKUP uses a RACF utility to copy the RACF definitions to an output data set (RACF internal format) used in job ICMMDATA.</li> <li>2. Job ICMMDATA converts the output file from ICMMBKUP to a sequential output file used in job ICMMSORT.</li> <li>3. Job ICMMSORT sorts the RACF definitions so they can be used as input to job ICMMCOMP.</li> </ol>
ICMMUSTB	Extracts list of users currently defined to Content Manager.

Member in SICMINS1	Purpose
ICMMCOMP	<p>Creates a delta list of differences between users defined in Content Manager and RACF using jobs mentioned above. The output is a list of users to be added to or deleted from CMExtract user information from RACF and sort it.</p> <p>The output file from this job <i>should be reviewed and modified as needed</i>.</p> <p>The format of each line in the file is fairly simple:</p> <ul style="list-style-type: none"> <li>▶ UserID (1 to 8 characters)</li> <li>▶ one space</li> <li>▶ The letter “A” for users to be added, the letter “D” for users to be deleted</li> <li>▶ Binary ‘00’x to fill record to 37 bytes</li> </ul>
ICMMDFUR	<p>Based on output file from ICMMCOMP, users are added to or deleted from Content Manager. You provide additional information to the user definition through the input data set DEFTXT. The input is supplied through a number of input lines as illustrated in Example 5-2 on page 118 and Example 5-3 on page 119. Refer to “Specifying ACLs and privilege sets for imported users” on page 116.</p>
ICMMCACL	<p>Recompiles the Content Manager ACLs to remove deleted users from the ACLs.</p>

**Important:** The values, set for privilege set and ACL, that are specified through the DEFTXT data set, are used for *all* users added during one run of the import process. If you need to assign different values to various groups of users, you must split the list of users up into multiple batches and run the import job for each group of users. Each of these jobs can then have different parameters set through the DEFTXT input.

The users are added to Content Manager without a value in the password field. They will only be able to log on to Content Manager if you are using the security exits that validate the user ID and password via RACF or a similar method. There is a way to get around this limitation. Refer to 5.2.10, “Copy encrypted password from sample user” on page 124 for a discussion of this topic.

## 5.2.6 Specifying ACLs and privilege sets for imported users

When adding users to Content Manager via the ICMMDFUR job, there are a number of additional values in the user definition that you need to set. They include:

- ▶ Domain ID for the user
- ▶ Privilege sets for the user
- ▶ ACL for the user
- ▶ Default Resource Manager in which the user stores documents
- ▶ Default collection the user uses on Resource Manager

Normally, these values are entered from the system administration client. Figure 5-1 and Figure 5-2 show the screens where you enter this type of information.

Figure 5-1 System administration client: User details for maximum privilege set

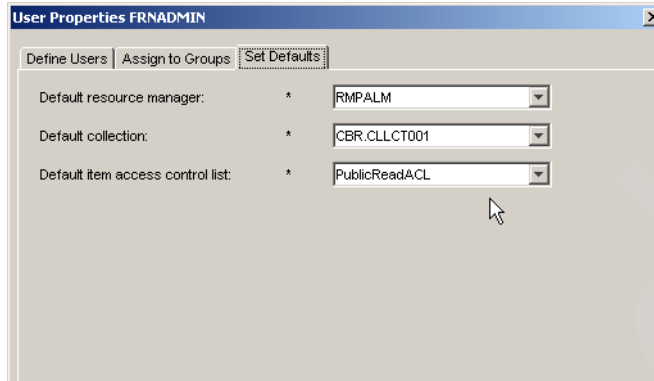


Figure 5-2 System administration client: Defaults for RM-related options

The values for fields “Privilege Set”, “Grant Privilege Set”, and “Default item access control list” can be specified through the DEFTXT input file when you run the batch job.

When providing these values through data set DEFTXT of job ICMMDFUR, you need to use the internal numeric values used by Content Manager and not the descriptive names displayed in the Content Manager system administration client.

To obtain the internal numeric values for these fields, you need to run some SQLs to extract these values from the Content Manager DB2 tables. Examples of the SQLs you need to run to obtain these values are shown in 5.2.7, “Sample SQLs to extract information from Content Manager” on page 119.

Example 5-2 shows the input you need to provide to the ICMMDFUR job.

---

*Example 5-2 Input to job ICMMDFUR*

---

```
//DEFTXT DD *
?DOMAINID?
?USERPRIVSETCODE?
?GRANTPRIVSETCODE?
?DEFAULTITEMACLCODE?
?RMCODE?
?SMSCOLLECTION?

/*
```

---

Based on the definitions in our system, Example 5-3 shows the actual input values to job ICMMDFUR.

*Example 5-3 Sample input to job ICMMDFUR using values from Content Manager*

---

```
//DEFTXT DD *
1000
5
11
4
1
1
/*
```

---

### 5.2.7 Sample SQLs to extract information from Content Manager

The input data, used by the ICMMDFUR job, as shown in Example 5-3, must be in the form of the internal numeric values from Content Manager. We include sample SQLs that you can use to:

- ▶ Extract information on privilege sets.
- ▶ Extract information on ACL codes.
- ▶ Extract information on Resource Manager.
- ▶ Extract information on Resource Manager collections.
- ▶ Extract information on domains.

**Important:** For all the sample SQLs, you must modify the schema name IFVTW to the one you use for your Library Server tables.

The SQLs can be run either from a DB2 command window on a Windows workstation or from SPUFI under TSO.

#### Extract information on privilege sets

Example 5-4 contains a sample SQL that extracts information on privilege sets from Content Manager.

*Example 5-4 Sample SQL to extract information on privilege sets*

---

```
--
-- Generate list of PRIVSET codes
-- You must change the schema name "ifvtw" to match your environment.
--
select KeywordClass, LanguageCode, KeywordCode,
       KeywordName PrivsetName, KeywordDescription PrivsetDescription
from ifvtw.icmstnlkeywords
where keywordclass = 11
order by languagecode, keywordcode;
```

---

The output is shown in Example 5-5. Use the value shown in column KEYWORDCODE to replace the lines saying “?USERPRIVSETCODE?” and “?GRANTPRIVSETCODE?” in Example 5-2 on page 118.

*Example 5-5 Sample list of privilege set code*

List of Privset Codes:

KEYWORDCLASS	LANGUAGECODE	KEYWORDCODE	PRIVSETNAME
11	ENU	0	PrivilegeSet
11	ENU	1	AllPrivs
11	ENU	2	NoPrivs
11	ENU	3	SysAdminCM
11	ENU	4	SysAdminSuper
11	ENU	5	ClientUserCreateAndDelete
11	ENU	6	ClientUserReadOnly
11	ENU	7	UserDB2Connect
11	ENU	8	UserDB2TrustedConnect
11	ENU	9	SysAdminEIP
11	ENU	10	ClientUserEdit
11	ENU	11	ClientUserAllPrivs
11	ENU	12	SysAdminSubDomainCM
11	ENU	13	LibrarianInfoMining
11	ENU	14	UserInfoMining
11	ENU	15	SysAdminSubDomainEIP

16 record(s) selected.

## Extract information on ACL codes

Example 5-6 contains a sample SQL that extracts information on ACL codes from Content Manager.

*Example 5-6 Sample SQL to extract information on ACL codes*

```
--
-- Generate list of ACL codes
-- You must change the schema name "ifvtw" to match your environment.
--
select KeywordClass, LanguageCode, KeywordCode,
       KeywordName ACLName, KeywordDescription ACLDescription
from ifvtw.icmstnlkeywords
where keywordclass = 13
order by languagecode, keywordcode;
```

Sample output of the SQL is shown in Example 5-7. Use the value shown in column KEYWORDCODE to replace the lines “?DEFAULTITEMACLCODE?” in Example 5-2 on page 118.

*Example 5-7 Sample list of ACL codes*

List of ACL codes:

KEYWORDCLASS	LANGUAGECODE	KEYWORDCODE	ACLNAME
13	ENU	0	ACL
13	ENU	1	SuperUserACL
13	ENU	2	NoAccessACL
13	ENU	3	PublicReadACL
13	ENU	4	DocRouteACL

5 record(s) selected.

**Extract information on Resource Manager**

Example 5-8 contains a sample SQL that extracts information on Resource Managers from Content Manager.

*Example 5-8 SQL to extract information on Resource Managers*

```
--
-- List Resource Managers
-- You must change the schema name "ifvtw" to match your environment.
--
select RMCode, RMName
from ifvtw.ICMSTResourceMgr
order by RMCode;
```

Sample output of the SQL is shown in Example 5-9. Use the value shown in column RMCODE to replace the lines “?RMCODE?” in Example 5-2 on page 118.

*Example 5-9 Sample list of Resource Managers*

List of Resource Managers:

RMCODE	RMNAME
0	reserved
1	RM390IFVTW

2 record(s) selected.

## Extract information on Resource Manager collections

Example 5-10 shows a sample SQL that extracts information on collections from Content Manager.

### *Example 5-10 SQL to extract information on collections*

---

```
--
-- List Collections
-- You must change the schema name "ifvtw" to match your environment.

--
select COL.RMCode, substr(RM.RMName,1,10) RMName,
       COL.SMSCollcode, COL.SMSCollname
from ifvtw.ICMSTCollname COL
left join ifvtw.ICMSTResourceMgr RM
on COL.RMcode = RM.RMcode
order by COL.RMCode, COL.SMSCollcode;
```

---

Sample output of the SQL is shown in Example 5-11. Use the value shown in column SMSCOLLCODE to replace the lines "?SMSCOLLECTION?" in Example 5-2 on page 118.

### *Example 5-11 Sample list of collections*

---

List of Collections:

RMCODE	RMNAME	SMSCOLLCODE	SMSCOLLNAME
1	RM390IFVTW	0	preFetch Reserved1
1	RM390IFVTW	1	CLLCT001
1	RM390IFVTW	2	CLLCT003

5 record(s) selected.

---

## Extract information on domains

Example 5-12 contains a sample SQL that extracts information on domains from Content Manager.

### *Example 5-12 SQL to extract information on Domains*

---

```
--
-- List Domains
-- You must change the schema name "ifvtw" to match your environment.
--
select DOM.DomainID, DN.KeywordName DomainName, DN.LanguageCode,
       DN.KeywordDescription DomainDescription
from ifvtw.ICMSTAdminDomains DOM
left join ifvtw.ICMSTNLSKeywords DN
```



```
on DN.KeywordCode = DOM.DomainID
and DN.KeywordClass = 17
order by DOM.DomainID, DN.LanguageCode;
```

---

Sample output of the SQL is shown in Example 5-13. Use the value shown in column DOMAINID to replace the lines “?DOMAINID?” in Example 5-2 on page 118.

*Example 5-13 Sample List of Domains*

---

List of Domains:

DOMAINID	DOMAINNAME	LANGUAGECODE	DOMAINDESCRIPTION
1	SuperDomain		Defined domain for super system administration.
2	PublicDomain		Defined domain for Public access.
1000	DefaultDomain		Defined default domain for sub-administration.
3 record(s) selected.			

---

## 5.2.8 Extract user list and information from Content Manager

Example 5-14 shows you how to obtain a list of users defined on your system and obtain each user’s settings such as ACL and privilege sets from a sample SQL.

*Example 5-14 SQL to extract information on users and their current settings*

---

```
--
-- List users and some of their info (Privsets, ACLs, etc)
-- You must change the schema name “ifvtw” to match your environment.
--
select distinct USR.UserId
, USR.UserPrivsetCode, UP.KeywordName UserPrivset
, USR.GrantPrivsetCode, UG.KeywordName GrantPrivset
, USR.DfltACLcode, ACL.KeywordName DfltACLcode
, USR.DomainID, DN.KeywordName DomainName
, USR.DfltRMcode, substr(RM.RMname,1,10) DfltRM
from ifvtw.ICMSTUsers USR

left join ifvtw.ICMSTNLSKeywords UP
on USR.UserPrivsetCode = UP.Keywordcode
and UP.KeywordClass = 11

left join ifvtw.ICMSTNLSKeywords UG
on USR.GrantPrivsetCode = UG.Keywordcode
and UG.KeywordClass = 11

left join ifvtw.ICMSTNLSKeywords ACL
```

```
on USR.DfltACLcode = ACL.Keywordcode
and ACL.KeywordClass = 13

left join ifvtw.ICMSTResourceMgr RM
on USR.DfltRMcode = RM.RMcode

left join ifvtw.ICMSTNLSKeywords DN
on USR.DomainID = DN.KeywordCode
and DN.KeywordClass = 17

order by USR.UserId;
```

Example 5-15 shows the output of the SQL. Note, the columns with the numeric values have been left out in this example to simplify the display here.

Example 5-15 List of users and their various settings

List of users and their settings:

USRID	USERPRIVSET	GRANTPRIVSET	DFLTACLCODE	DOMAINNAME	DFLTRM
ICMPUBLIC	AllPrivs	AllPrivs	SuperUserACL	PublicDomain	reserved
ICMUSR2	UserDB2Connect	UserDB2Connect	PublicReadACL	DefaultDomain	RM390IFVTW
IFVTW	AllPrivs	ClientUserReadOnly	PublicReadACL	SuperDomain	RM390IFVTW

3 record(s) selected.

### 5.2.9 Alternative ways of providing input to user import job

Alternatively, you may want to create the input data set for user import job, ICMMDFUR, a source other than RACF.

The alternative input data set must be a sequential data set with records in the following format:

- ▶ 1 to 8 character user ID
- ▶ One space (in example: x'40')
- ▶ One character action value:
  - “D” for delete the user.
  - “A” for add the user.
- ▶ Binary zeroes to pad record to 37 bytes

### 5.2.10 Copy encrypted password from sample user

The initial password for new users, added through the import program, cannot be set. You may want to set it in case you are not using security exits.

As an example, if you need to load a demo or test system and would like to set an initial password for all the test users, you may use the following method:

1. Define a sample user and password using the system administration client. In our example, we use BASEDEMO as the user ID. The password of the user is saved in the Content Manager table, ICMSTUSERS, in an encrypted format.
2. Add the group of users to the system using the import program. You could, for example, import a list of test users using the name “DEMONnn” where *nnn* is a sequence number.
3. Use an SQL statement to copy the password from the sample user to the imported users. To update a list of users in one SQL statement, use a naming convention such as DEMONnn for the test users and then use wildcard in your SQL statement to get the list of users for whom you want to update passwords.

Example 5-16 shows a sample SQL that sets the password of all users whose IDs start with DEMO, to the password of the user, BASEDEMO.

*Example 5-16 Sample SQL to set password of users from an existing password*

---

```
Update ifvtw.ICMSTUSERS
set Password =
(select Password from ifvtw.ICMSTUsers where UserId = 'BASEDEMO')
where UserId like 'DEMO%';
```

---

When calling from your system, change the schema name to match your environment and set the value of the userIDs to the relevant values.

## 5.3 ACL user exit routines overview

Content Manager document security is implemented through the use of access control lists (ACLs). You can replace this built-in security by activating the exits described in this section.

**Note:** If you activate the ACL user exit routines, the Content Manager ACLs will not be used at all.

Once you use the exits, they replace all ACL-related functions on this Library Server. You must implement the full security model in this case.

There are two exits involved in this process:

- ▶ ICMACLPPrivExit
- ▶ ICMGenPrivExit

## 5.4 ICMACLPriExit

*ICMACLPriExit* enables you to validate every single request the users perform against the Library Server. Examples of the requests include searching, storing, and deleting documents.

Before you start to activate this exit, we strongly recommend you evaluate the existing functionality that Content Manager provides through its built-in implementation of the ACLs. The security model available through the ACLs is very comprehensive. It should support most business requirements.

If you decide to use the security exits, remember that you have to evaluate every single request performed against the Library Server and you should have logic in place that determines whether to allow or reject requests for any given user under all circumstances.

### 5.4.1 ICMACLPriExit definition in DB2

ICMACLPriExit is defined as a DB2 User Defined Function (UDF) that invokes the external program, ICMACLXT.

Example 5-17 contains the source code for ICMACLPriExit.

*Example 5-17 DB2 UDF ICMACLPriExit*

---

```
1. create function IFVTW.ICMACLPRIEXIT
2. (
3. VARCHAR(33),
4. VARCHAR(32670),
5. varchar(151),
6. INTEGER,
7. VARCHAR(20),
8. INTEGER,
9. CHAR(26),
10. INTEGER,
11. INTEGER
12. )
13. RETURNS INTEGER
14. NO COLLID
15. WLM ENVIRONMENT WLMICM1
16. PROGRAM TYPE SUB
17. EXTERNAL NAME ICMACLXT
18. LANGUAGE C
19. PARAMETER STYLE DB2SQL
20. FENCED
21. STAY RESIDENT YES
22. NO SQL;
```

---

The ICMACLPriExit UDF is defined in the installation job ICMMLSCR.

In line 1 of the source code, the name of the UDF is defined. From line 3 to line 11, 9 input parameters to the UDF are declared. In line 13, the type of return value is declared. In line 15, we specify which Workload Manager (WLM) to execute. In line 17, the name of the external module to be called is defined.

## 5.4.2 Activating ICMACLPriExit

The ACL exit can be activated via the Content Manager system administration client. From the client's Library Server configuration screen, under the Feature tab, as shown in Figure 5-3, you can check the box "Enable ACL User Exit" to enable it.

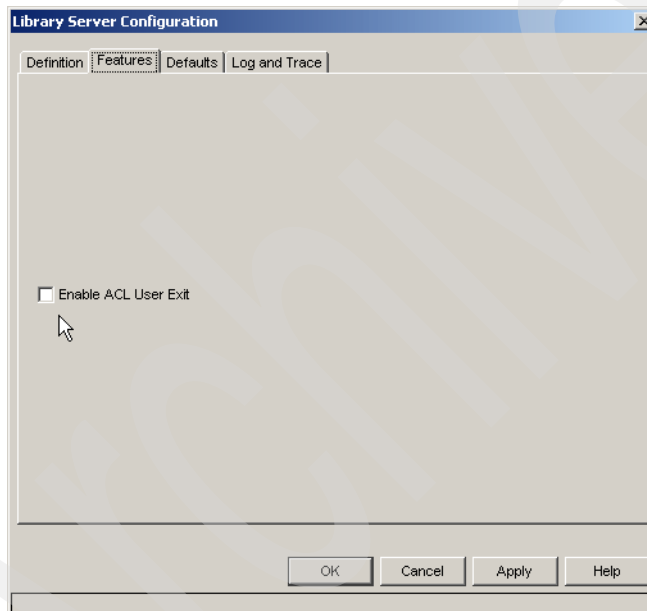


Figure 5-3 Enable ACL user exit program from the system administration client

In Content Manager V8.2, the activation or deactivation of this option required regeneration of all Content Manager access modules. Depending on the settings used when the Content Manager batch jobs were submitted, additional manual steps were required to finalize the changes. If you did not regenerate the access modules, you could not access the system.

In Content Manager V8.3, we no longer use access modules. This eliminates the access module regeneration step. Instead, Content Manager structures SQL

statements in a table and uses them to build dynamic SQL statements for accessing item type tables.

### 5.4.3 Sample ACL exit program and parameter list

In installation library xxx.SICMSAM1, there is a sample exit program in member ICMACLXT.

This sample program first shows the parameter list passed to the program. See Example 5-18 for the declaration of the parameters passed.

*Example 5-18 Declaration of parameters passed to program ICMACLXT*

---

```
void SQL_API_FN ICMACLPrivExit(  
    SQLUDF_CHAR      *pszUserID,          /* input */  
    SQLUDF_CHAR      *pszApplicationID, /* input */  
    SQLUDF_CHAR      *pszHostname, /* input */  
    SQLUDF_INTEGER    *APIAction,         /* input */  
    SQLUDF_CHAR      *pszProcedureName, /* input */  
    SQLUDF_INTEGER    *InfoType,          /* input */  
    SQLUDF_CHAR      *ItemID,             /* input */  
    SQLUDF_INTEGER    *ViewID,            /* input */  
    SQLUDF_INTEGER    *pIgenPrivCode,     /* input */  
    SQLUDF_INTEGER    *pIrc,              /* output */  
    SQLUDF_SMALLINT   *pszUserID_ind,  
    SQLUDF_SMALLINT   *pszApplicationID_ind,  
    SQLUDF_SMALLINT   *pszHostname_ind,  
    SQLUDF_SMALLINT   *APIAction_ind,  
    SQLUDF_SMALLINT   *pszProcedureName_ind,  
    SQLUDF_SMALLINT   *InfoType_ind,  
    SQLUDF_SMALLINT   *ItemID_ind,  
    SQLUDF_SMALLINT   *ViewID_ind,  
    SQLUDF_SMALLINT   *pIgenPrivCode_ind,  
    SQLUDF_SMALLINT   *pIrc_ind,  
    SQLUDF_TRAIL_ARGS_ALL  
)
```

---

You can match the parameter list against the definition of the DB2 UDF in Example 5-17 on page 126. To find more information on DB2 User Defined Function (UDF), refer to the following documentation:

- ▶ *DB2 V7 z/OS Application Programming and SQL Guide*, SC26-9933
- ▶ *DB2 V7 z/OS SQL Reference*, SC26-9944
- ▶ *DB2 UDB V8 SQL Reference Volume 1*, SC09-4844
- ▶ *DB2 UDB V8 SQL Reference Volume 2*, SC09-4845

These publications can be downloaded from the following IBM Web site:

<http://www.ibm.com/servers/eserver/zseries/zos>

When you look at the parameter definition in the sample program, you should notice the following points about the parameter list:

- ▶ The first nine parameters are the data values passed to the program matching the parameters specified in the UDF definition. These are the input values passed from Content Manager.

The tenth parameter is the return code, as specified in the “RETURNS INTEGER” line of the UDF, that is set by the exit program and returned to the caller.

- ▶ The next ten parameters are indicator variables matching the sequence of the parameters. For example, the first indicator variable corresponds to parameter one, and the second indicator matches parameter two, and the tenth indicator matches the return code returned by the program.

The indicators are used by DB2 to indicate whether one of the passed or returned parameters is a NULL value. If the indicator value is set to -1 (minus one), then the value in the corresponding parameter should be NULL, and the actual content of the parameter variable should be undefined.

- ▶ The last parameter in the list is status information passed from and to DB2.

Table 5-3 lists parameters that are passed to the exit program.

*Table 5-3 Parameters passed to the ICMACLPriExit program ICMACLXT*

Parameter	Description
UserId	Active Content Manager user ID.
ApplicationID	Currently not used. Reserved for future usage.
Hostname	Host name of the client.
APIAction	Type of function to be performed. A list of the numeric values can be passed and their meanings are as follows: <ul style="list-style-type: none"><li>▶ 0: Retrieve</li><li>▶ 2: Update</li><li>▶ 3: Delete</li><li>▶ 4: Future use</li></ul>
ProcedureName	The name of the Library Server stored procedure requesting the ACL validation.
InfoType	Type of item on which the request is working: <ul style="list-style-type: none"><li>▶ 1: Items</li><li>▶ 2: Views</li><li>▶ 3: Item types or item type views</li></ul>

Parameter	Description
ItemID	ItemID of the item on which the action is working.
ViewID	Item type ID, item type view ID, or view ID. This is the internal numeric identification of the item type on which the process is working.
GenPrivCode	The general privilege required to perform the current action.
RC	Return code returned to caller. If you return 0, this grants the request to continue. Any other values decline the request and Content Manager returns an error to the caller.

It is up to your program to interpret the request. Based on the passed information, your program determines whether the user is authorized to perform a specific request or not.

#### 5.4.4 Call types

Table 5-19 describes the various types of calls that are passed to the exit.



Example 5-19 Overview of parameter sets passed to ACL exit

Action	ItemID	ViewID	Procedure name	Comments
0 (Retrieve)	n/a	n/a	n/a	Retrieve server definitions
	available	300 to 304	GetItem	Retrieve resource items
	available	>=1000	GetItem	Retrieve items
	available	>=1000	ICMGETLINKEDITEMS	Retrieve linked items
	n/a	300 to 304 and >=1000	openQueryCursor	Items returned in a Search resultset
	available	n/a	ICMGetWork	Document routing
	available	n/a	ICMGetWorkList	Document routing
1 (Insert)	available	300 to 304	CreateItem	Create resource item - either new or as part of update/replace operation
	available	>=1000	CreateItem	Create new item type instance, either root or child item
2 (Update)	available	300 to 304	UpdateItem	Update resource item
	available	>=1000	UpdateItem	Update item attributes
3 (Delete)	available	0	DeleteItem	Delete part. This can either be a delete as such or part of an update item process
	available	300 to 304	DeleteItem	Delete resource item
	available	>=1000	DeleteItem	Delete root or child item
4 (Miscellaneous)	maybe available	all	ICMCheckPrivForItemACL	Miscellaneous such as Checkout item, or Checkin item

## 5.5 ICMGenPrivExit

The purpose of this exit is to enable you to verify a user's general privilege to perform a certain function.

## 5.5.1 Activating ICMGenPrivExit

You activate ICMGenPrivExit at the same time you activate ICMACLPPrivExit through Content Manager system administration client. Refer to 5.4.2, “Activating ICMACLPPrivExit” on page 127 to read about how to activate the exit.

Note, there is no option to activate the general privilege exit on its own. You either activate both or neither.

If you wish to only use ICMACLPPrivExit and continue to use Content Manager’s internal validation of the general privileges based on the user’s profile, you can do this by not making the ICMGENXT program available in any STEPLIB of the WLM where the Library Server runs.

## 5.5.2 Sample exit program and parameter list

You implement this exit through module ICMGENXT. The source code for the sample program is available in member ICMGENXT in library SICMSAM1.

Example 5-20 shows the parameters that are passed to the program.

*Example 5-20 Declaration of parameter list passed to program ICMGENXT*

---

```
long ICMGenPrivExit(  
    char *pszUserID,  
    char *pszApplicationID,  
    char *pszHostname,  
    long *plGenPrivCode,  
    char *pszProcedureName,  
    long *plCMDecision,  
    long *plCMDecReason,  
    long *plRC)
```

---

Note the difference to the format of the parameter list passed to the ICMACLPXT program. The ICMGenPrivExit program is not defined as nor called as a DB2 UDF; rather, it is a direct program to program call. There are no indicator variables nor DB2-related parameters in the definition of the parameter list.

## 5.6 Writing exits in non-C programming languages

By default, the C programming language is used to write the Content Manager exits. You may need to write the ACL exit in a language other than C. You may also need to perform SQL calls in the exit program to access your security tables in DB2 or possibly the Content Manager tables themselves to get additional information.

One way to do this is to redefine the UDF for the ICMACLPriExit to call a PL/I program. Using the same techniques, you could make use of a COBOL program.

### 5.6.1 Redefining UDF

In order to call a program type other than the default C program shipped in the installation material, you must change the UDF definition.

For some of the parameters associated with an UDF, you can normally use the DB2 ALTER function to change the UDF. However, when changing the fundamental parts of the definition such as the programming language and the NOSQL option, you have to first drop the existing definition and then redefine the UDF.

DB2 keeps track of those programs which use UDFs. Before dropping the UDF, you should list any dependencies on the UDF. Dropping it may require you to perform a REBIND of the involved packages.

You can list the dependencies by calling the SQL as shown in Example 5-21. Replace the value ITS08ADM with the schema name you use for your Library Server database.

*Example 5-21 SQL to list dependencies on a UDF*

---

```
Select BQUALIFIER, BNAME, BTYPE, DCOLLID, DNAME
From SYSIBM.SYSPACKDEP
Where BNAME = 'ICMACLPRIEXIT'
and BQUALIFIER = 'ITS08ADM';
```

---

The output of the sample SQL call is shown in Example 5-22.

*Example 5-22 List of dependencies on UDF ICMACLPriExit*

---

BQUALIFIER	BNAME	BTYPE	DCOLLID	DNAME
ITS08ADM	ICMACLPRIEXIT	F	ITS08PK	ICMPLSPC

1 record(s) selected.

---

In this example, the package ICMPLSPC is dependent on the UDF. You need to rebind this package after redefining the UDF.

The easiest thing to rebind the package after redefining the UDF may be to submit the installation job ICMMLSD for the Library Server you are working with, since you have already tailored this job to perform all the necessary binds for your system.

## 5.6.2 Sample PL/I program with parameter list

Example 5-23 contains a sample PL/I program that shows the format of the parameter list passed by DB2 to a UDF written in PL/I as well as a few statements to validate a request. The main job is to determine how and what to validate according to your business requirements.

*Example 5-23 Sample PL/I program for Content Manager exit ACLPrivExit*

---

```
*PROCESS SYSTEM(MVS);
/*-----*/
/* Link with "DB2 LANGUAGE INTERFACE MODULE" DSNRLI */
/*-----*/
ACLPRIV: PROC (
    USERID
    ,APPLICATIONID
    ,HOSTNAME
    ,APIACTION
    ,PROCEDURENAME
    ,INFOTYPE
    ,ITEMID
    ,VIEWID
    ,GENPRIVCODE
    ,RC

    ,IND_USERID
    ,IND_APPLICATIONID
    ,IND_HOSTNAME
    ,IND_APIACTION
    ,IND_PROCEDURENAME
    ,IND_INFOTYPE
    ,IND_ITEMID
    ,IND_VIEWID
    ,IND_GENPRIVCODE

    ,IND_RC

    ,UDF_SQLSTATE
    ,UDF_NAME
    ,UDF_SPEC_NAME
    ,UDF_DIAG_MSG
)
OPTIONS(MAIN NOEXECOPS REENTRANT);

/*----- */
/* Data passed by Content Manager in call to UDF */
/*----- */
DCL USERID          VAR  CHAR(33);
DCL APPLICATIONID   VAR  CHAR(32670);
```

```

DCL HOSTNAME          VAR  CHAR(151);
DCL APIACTION         FIXED BIN(31);
DCL PROCEDURENAME     VAR  CHAR(20);
DCL INFOTYPE          FIXED BIN(31);
DCL ITEMID            CHAR(26);
DCL VIEWID            FIXED BIN(31);
DCL GENPRIVCODE       FIXED BIN(31);
DCL RC                FIXED BIN(31);

/* ----- */
/* Indicator variables set/used by DB2 to indicate */
/* NULL values in corresponding parameter */
/* ----- */
DCL IND_USERID        FIXED BIN(15);
DCL IND_APPLICATIONID  FIXED BIN(15);
DCL IND_HOSTNAME      FIXED BIN(15);
DCL IND_APIACTION     FIXED BIN(15);
DCL IND_PROCEDURENAME  FIXED BIN(15);
DCL IND_INFOTYPE      FIXED BIN(15);
DCL IND_ITEMID        FIXED BIN(15);
DCL IND_VIEWID        FIXED BIN(15);
DCL IND_GENPRIVCODE   FIXED BIN(15);
DCL IND_RC            FIXED BIN(15);

DCL UDF_SQLSTATE      CHAR (5);
DCL UDF_NAME          VAR  CHAR (137);
DCL UDF_SPEC_NAME     VAR  CHAR (128);
DCL UDF_DIAG_MSG      VAR  CHAR (70);

DCL TIME              BUILTIN;
DCL INDEX             BUILTIN;
DCL SUBSTR            BUILTIN;
DCL ADDR              BUILTIN;

/* ----- */
/* Log passed parameters to SYSPRINT of the WLM where */
/* program executes */
/* ----- */
PUT SKIP;
PUT SKIP LIST (TIME, '>>> CM ACLPRIVEXIT program');
PUT SKIP DATA (USERID          );
PUT SKIP DATA (APPLICATIONID    );
PUT SKIP DATA (HOSTNAME        );
PUT SKIP DATA (APIACTION       );
PUT SKIP DATA (PROCEDURENAME    );
PUT SKIP DATA (INFOTYPE        );
PUT SKIP DATA (ITEMID          );
PUT SKIP DATA (VIEWID          );
PUT SKIP DATA (GENPRIVCODE     );

```

```

/* ----- */
/* Add the logic you wish to perform for each function */
/* such as READ, UPDATE, DELETE as determined by the */
/* value of APIACTION */
/* ----- */
SELECT (APIACTION);

/* ----- */
/* Is this a READ request? */
/* Check VIEWID: */
/* if it's less than 1000 it's a resource or wrkflw */
/* else it may be a document we need to check access */
/* to by validating the USERID, etc. */
/* ----- */
WHEN (0) DO;
  IF VIEWID < 1000 THEN RETURN;
  IF SUBSTR(PROCEDURENAME,INDEX(PROCEDURENAME,'.')+1)
    = 'ICMGETDOCTOC'
  THEN DO;
    /* add your validation here... */
    /* To REJECT request do: "RC = -1;" */
    /* To ALLOW request do: "RC = 0;" */
    END;
  ELSE RETURN;
END;

/* ----- */
/* Is this a CREATE request? */
/* ----- */
WHEN (1) DO;
  /* Add logic to handle CREATE requests */
  END;

/* ----- */
/* Is this an UPDATE request? */
/* ----- */
WHEN (2) DO;
  /* Add logic to handle UPDATE requests */
  END;

/* ----- */
/* Is this a DELETE request? */
/* ----- */
WHEN (3) DO;
  /* Add logic to handle DELETE requests */
  END;

/* ----- */

```

```

/* Ignore the rest */
/* ----- */
    OTHERWISE RETURN;

END; /* SELECT complete */

/* ----- */
/* Depending on your findings above you control access */
/* by setting the return code in return variable RC : */
/* RC = 0 will allow the request to proceed */
/* RC = -1 will cause the request to be rejected */
/* ----- */
EXITPGM:

END ACLPRIV;

```

---

### 5.6.3 Linking exit program

The exit program that was shown in Example 5-23 is invoked by the DB2 UDF and runs in the designated WLM.

There are special linkage requirements for programs called by DB2. You must include the correct DB2 stub as shown in Example 5-24.

*Example 5-24 Linking the exit program with DSNRLI*

---

```

//LINK.SYSIN DD *
INCLUDE SYSLIB(DSNRLI)
INCLUDE PLKLIB(ACLPRIV)
NAME ACLPRIV(R)
/*

```

---

To get additional information on linking the program for use under WLM, refer to the following documentation:

- ▶ *DB2 UDB V7 for z/OS Application Programming and SQL Guide*, SC26-9933
- ▶ *DB2 UDF V8 for z/OS Application Programming and SQL Guide*, SC18-7415

You can download them from the following IBM Web site:

<http://www.ibm.com/software/data/db2/zos>

## 5.7 Additional options with DB2 triggers

If you have additional requirements to handle security in your Content Manager system, you may use DB2 triggers on Content Manager DB2 tables to provide the additional functionalities.

**Disclaimer:** Use DB2 triggers in Content Manager system at your own risk. It is not the responsibility of the IBM DB2 Content Manager Technical Support Team nor the IBM ITSO Redbook Team to provide support for issues related to your DB2 triggers for a Content Manager system. We provide the information here as a possible option only.

### 5.7.1 DB2 trigger overview

You can define a DB2 trigger on a table and specify whether or not it should be invoked before data is inserted, updated, or deleted from the table. The trigger has access to the data DB2 is planning to insert, update, or delete from the table. The trigger has the option to allow or reject the request after evaluating the request.

The DB2 trigger is defined much the same way as a UDF. It can be defined with SQL statements directly in the trigger definition or it can call an external program that implements complex business logic.

In the case where the trigger rejects the request by setting a return code and a SQLSTATE code, DB2 returns these codes to the caller. The trigger can use different SQLSTATE codes to reflect the reason for a denial of the request so the caller can interpret the cause of rejection.

### 5.7.2 Using DB2 triggers with Content Manager

One example of using DB2 triggers with Content Manager is validating certain key field values before storing a document in Content Manager. If for security reasons, you cannot rely on this getting done on the client side; you may choose to monitor the data before it is saved to Content Manager.

One preferable way to do this could be through an exit such as ICMACLPriExit where you could monitor the create item calls. The problem with this approach is that the ACL exit does not have access to the key values of the new item.

An alternative approach would be to add a trigger on the root item table that validates inserting, updating, and deleting requests because one of the first steps Content Manager does when creating a new item in the Library Server is to create a row in the root item table.



The DB2 trigger may then be based on the user ID of the requestor and the value of a selected index field verifying the request. The DB2 trigger can then either permit or reject the request.

### 5.7.3 Example of a simple trigger calling a UDF

Example 5-25 shows you how to define a simple DB2 trigger on a Content Manager root item table.

*Example 5-25 Creating a simple DB2 trigger on Content Manager's root item table*

---

```
1. Create Trigger Trig01020
2. No Cascade
3. Before Insert
4. On IFVTW.ICMUT01020001
5. Referencing New As N
6. For Each Row Mode DB2SQL
7. When (N.ATTR0000001042 > 1000)
8. Signal SQLSTATE '75000' ('Invalid value for attribute xxx')
```

---

Line 1 of Example 5-25 defines the name of the DB2 trigger. Line 3 specifies that the trigger is to be called before any row is inserted. Line 4 specifies the name of the root component table. Line 7 checks the value for the attribute in column ATTR0000001042. If line 7 is true, it continues to line 8, which rejects the insert with invalid values by setting the SQLSTATE and return with the error message.

Before using the example, remember to replace the DB2 table name and the name of the attribute value to match the actual definitions on your system.

In Appendix E, “List and validate Content Manager definitions via SQL” on page 381, we provide SQLs that help you generate a list of the item type names and attribute names you defined in Content Manager that are mapped to the actual DB2 table and column names.

### 5.7.4 Calling a program from a trigger

If you need to implement a more complex business logic to evaluate an update request to a table, you can call an external program and pass the necessary parameters to it. For example, you can use the following steps to achieve this:

1. Define the UDF with the parameters and specify the program to be called.
2. Define a DB2 trigger on a root component table that calls the UDF.
3. Receive the parameters in a program and pass back a return code to the UDF.

## Define DB2 UDF

Example 5-26 shows you how to define the DB2 UDF.

*Example 5-26 Create the UDF calling external program*

---

1. Create Function **FVTFunctionA**(
  2. Char(32), Varchar(100) )
  3. Returns Integer
  4. WLM Environment **WLMFVT1**
  5. External Name **FVTPGM1**
  6. Language **PLI** Parameter Style DB2SQL
  7. Scratchpad Variant Fenced
  8. Final Call Null Call No SQL No External Action
- 

Line 1 of the example defines the name of the UDF. Lines 2 and 3 specify that the UDF expects two input parameters and returns an integer. Line 4 defines the Workload Manager in which the external program runs. You may specify the same WLM that is used by the Library Server. Line 5 specifies the name of the external program to call. This program must be made available to the WLM through a STEPLIB in the WLM procedure. In line 6, it specifies the programming language in which the called program is written. This example illustrates the use of a PL/I program. You can use a COBOL or Assembler program to accomplish this as well.

## Define a trigger on Content Manager root component table

Example 5-27 shows you how to define a DB2 trigger on the Content Manager root component table.

*Example 5-27 Create a trigger on root component table calling a UDF*

---

1. Create Trigger **Trig01020**
  2. No Cascade
  3. Before Insert
  4. On **IFVTW.ICMUT01020001**
  5. Referencing New As N
  6. For Each Row Mode DB2SQL
  7. When( **FVTFunctionA**(
  8. N.CreateUserID,
  9. N.ATTR0000001042)
  10. > 0)
  11. Signal SQLSTATE '75000' ('Not authorized to xxx...')
- 

In line 1 of the example, it defines the name of the trigger. In line 3, it specifies that the trigger is to be called before adding the new row to the table. In line 4, it specifies the name of the Content Manager root component table on which the trigger is created. In line 7, it calls the UDF defined previously. In line 8, it passes,

as the first parameter, the Content Manager user ID of the user requesting the insert of the new row. In line 9, it passes, as the second parameter, the attribute we need to validate. If we need to pass additional attribute values, update the UDF and the external program to support the additional parameters and pass them here. In line 10, it checks the return code returned by the UDF. In line 11, it sets the SQLSTATE to an error value if we got a non-zero return code from the UDF. This causes DB2 to reject the insert of the new row and the error is returned to the program that tried to insert it.

## Program called by UDF

As a last step, we show you a small sample program that is called by the UDF and illustrates the parameters that passed to the program.

The program logic can include any business logic that needs to be done. For example, you may want to validate the metadata for the attributes or a user's authority to use certain values. In case of invalid data, the program sets the return code, which is tested by the DB2 trigger that uses the UDF.

Example 5-28 contains the sample program.

*Example 5-28 Sample PL/I program called by a DB2 UDF*

---

```

*PROCESS SYSTEM(MVS);
/* ----- */
/* Define parameter list passed to program by DB2 based on the UDF */
/* definition. If you change the UDF definition (e.g. adding more */
/* parameters to be passed), you'll need to update the program */
/* definition as well! */
/* */
/* UDF_CREATEUSERID - first input parm, userid who creates item */
/* UDF_METADATA1 - second input parm, the key value */
/* UDF_RC - output parm, the return code */
/* UDF_CREATEUSERID_IND - indicator variable to detect NULL value */
/* UDF_METADATA1_IND - indicator variable to detect NULL value */
/* UDF_RC_IND - indicator variable to detect NULL value */
/* */
/* Remaining parameters are DB2-related information and the number */
/* of variables passed are depending on the optional parameters set */
/* in the UDF definition. */
/* ----- */

MYMAIN: PROC(
    UDF_CREATEUSERID,
    UDF_METADATA1,
    UDF_RC,
    UDF_CREATEUSERID_IND,
    UDF_METADATA1_IND,
    UDF_RC_IND,

```

```

    UDF_SQLSTATE, UDF_NAME, UDF_SPEC_NAME,
    UDF_DIAG_MSG, UDF_SCRATCHPAD,
    UDF_CALL_TYPE, UDF_DBINFO)
OPTIONS(MAIN NOEXECOPS REENTRANT);

DCL UDF_CREATEUSERID      CHAR(32);          /* Inp: userid          */
DCL UDF_METADATA1        CHAR(100) VARYING; /* Inp: metadata       */
DCL UDF_RC                BIN FIXED(15);      /* Out: Return Code     */

DCL UDF_CREATEUSERID_IND  BIN FIXED(15);      /* indicator for 1st parm */
DCL UDF_METADATA_IND     BIN FIXED(15);      /* indicator for 2nd parm */
DCL UDF_RC_IND            BIN FIXED(15);      /* indicator for result   */

DCL UDF_SQLSTATE          CHAR(5);           /* SQLSTATE returned to DB2 */
DCL UDF_NAME              CHAR(137) VARYING; /* Qualified function name */
DCL UDF_SPEC_NAME         CHAR(128) VARYING; /* Specific function name */
DCL UDF_DIAG_MSG          CHAR(70) VARYING; /* Diagnostic string */
DCL 01 UDF_SCRATCHPAD,    /* Scratchpad */
      03 UDF_SPAD_LEN     BIN FIXED(31),
      03 UDF_SPAD_TEXT    CHAR(100);
DCL UDF_CALL_TYPE          BIN FIXED(31);      /* Call Type */
DCL DBINFO                PTR;

/* CONSTANTS FOR DB2_ENCODING_SCHEME */
DCL SQLUDF_ASCII          BIN FIXED(15) INIT(1);
DCL SQLUDF_EBCDIC         BIN FIXED(15) INIT(2);
DCL SQLUDF_MIXED          BIN FIXED(15) INIT(3);
DCL 01 UDF_DBINFO         BASED(DBINFO),      /* Dbinfo */
      03 UDF_DBINFO_LLEN  BIN FIXED(15),      /* location length */
      03 UDF_DBINFO_LOC   CHAR(128),          /* location name */
      03 UDF_DBINFO_ALEN  BIN FIXED(15),      /* auth ID length */
      03 UDF_DBINFO_AUTH  CHAR(128),          /* authorization ID */
      03 UDF_DBINFO_CDPG, /* CCSIDs for DB2 for OS/390 */
      05 DB2_CCSIDS(3),
      07 R1 BIN FIXED(15), /* Reserved */
      07 DB2_SBCS BIN FIXED(15), /* SBCS CCSID */
      07 R2 BIN FIXED(15), /* Reserved */
      07 DB2_DBCS BIN FIXED(15), /* DBCS CCSID */
      07 R3 BIN FIXED(15), /* Reserved */
      07 DB2_MIXED BIN FIXED(15), /* MIXED CCSID */
      05 DB2_ENCODING_SCHEME BIN FIXED(31),
      05 DB2_CCSID_RESERVED CHAR(8),
      03 UDF_DBINFO_SLEN BIN FIXED(15),      /* schema length */
      03 UDF_DBINFO_SCHEMA CHAR(128),        /* schema name */
      03 UDF_DBINFO_TLEN BIN FIXED(15),      /* table length */
      03 UDF_DBINFO_TABLE CHAR(128),         /* table name */
      03 UDF_DBINFO_CLEN BIN FIXED(15),      /* column length */
      03 UDF_DBINFO_COLUMN CHAR(128),        /* column name */
      03 UDF_DBINFO_RELVER CHAR(8),          /* DB2 release level */

```

```

03 UDF_DBINFO_PLATFORM BIN FIXED(31), /* database platform */
03 UDF_DBINFO_NUMTFCOL BIN FIXED(15), /* # of TF cols used */
03 UDF_DBINFO_RESERV1 CHAR(24), /* reserved */
03 UDF_DBINFO_TFCOLUMN PTR, /* -> table fun col list */
03 UDF_DBINFO_APPLID PTR, /* -> application id */
03 UDF_DBINFO_RESERV2 CHAR(20); /* reserved */

/* ----- */
/* Add program code to do your checking of the metadata, e.g.: only user */
/* 'ADMINSJ' is allowed to create documents related to San Jose */
/* ----- */
IF UDF_METADATA1 = 'SAN JOSE' AND UDF_CREATEUSERID <> 'ADMINSJ'
THEN DO;
    UDF_RC = 99;
END;

```

---

### 5.7.5 Update Content Manager to handle SQL return codes

If you use a DB2 trigger in Content Manager V8.2 and the trigger rejects an update request by setting the return code and SQLSTATE, these values are not returned all the way through the API layer to the caller.

Content Manager V8.3 has been enhanced in this area. The Content Manager error message, return/reason code, and extended return/reason code that are returned to the caller will include the SQL return code and SQLSTATE values set by the trigger. Your program can evaluate the return/reason codes and present a meaningful message to the user.

You can also test these codes in an application program using the APIs. For more information on the Content Manager application programming interfaces, refer to the following publications:

- ▶ *IBM DB2 Content Manager Enterprise Edition V8.3: Application Programming Guide*, SC27-1347
- ▶ *IBM DB2 Content Manager Enterprise Edition V8.3: Application Programming Reference*, SC27-1348

## 5.8 Resource Manager exits

Resource Manager exits deal with pre- and post-processing of object storage and retrieval operations.

The Resource Manager exits are implemented through the programs listed in Table 5-4. They are defined in pairs of PRE and POST exits for a given function.

Table 5-4 List of Resource Manager exit programs

Exit program	Description
PRESTOR POSTSTOR	Executes before/after storing a new object.
PRERTRV POSTRTRV	Executes before/after retrieving an object.
PREREPL POSTREPL	Executes before/after replacing an existing object.
PRECHSM POSTCHSM	Executes before/after updating the SMS information for an object.
PREQUSM	Executes before querying the SMS information for an object.
PREDELT POSTDELT	Executes before/after deleting an object.
PREPFCH POSTPFCH	Executes before/after pre-fetching an object.

Refer to the following manual for more detailed information on exits:

- *IBM DB2 Content Manager for z/OS: Planning and Installing Your Content Management System*, GC18-7698

### 5.8.1 Parameters passed to Resource Manager exit programs

All calls to the exit programs by Content Manager pass the same parameter list. Some of the values can be modified by the PRE exit programs, others are for information only. In the POST exit program, you cannot modify any of the parameters.

Table 5-5 lists some of the parameters passed to the exit program. In installation data set xxx.SICMINS2, you can see a sample exit program in member ICMRMSP. The program includes the layout of the control block which shows you all of the parameters passed to the exit program.

Table 5-5 Parameters passed to Resource Manager exits

Parameter	Modifiable
OAM object name	no
ItemID	no
Transaction ID	no

Parameter	Modifiable
User ID of caller	no
Action type	no
Collection name	yes
Pre-fetch collection name	yes
New collection name	yes
New storage class	yes
New management class	yes
New retention period	yes

## 5.8.2 Activating Resource Manager exits

The ICMRMCONTROL table has one row of data controlling the activation of the exits. The table has a column per exit type controlling whether the exit is activated (value is 1) or deactivated (value is 0).

The columns and the exit controlled by these columns are shown in Table 5-6.

*Table 5-6 Columns in ICMRMCONTROL table and the exit that they control*

Column in DB2 table ICMRMCONTROL	Controls exit program
PRESTOREEXITPGM	PRESTOR
POSTSTOREEXITPGM	POSTSTOR
PRERTRVEXITPGM	PRERTRV
POSTRTRVEXITPGM	POSTRTRV
PREREPLEXITPGM	PREREPL
POSTREPLEXITPGM	POSTREPL
PRECHSMEXITPGM	PRECHSM
POSTCHSMEXITPGM	POSTCHSM
PREQUXMEXITPGM	PREQUXM
PREDELTEXTITPGM	PREDELT
POSTDELTEXTITPGM	POSTDELT

Column in DB2 table ICMRMCONTROL	Controls exit program
PREPFCHEXITPGM	PREPFCH
POSTPFCHEXITPGM	POSTPFCH

There is no option in the Content Manager system administration client to enable the exits. You have to update the DB2 table ICMRMCONTROL table using SQL executed via SPUFI or from a DB2CMD window on a workstation to enable or disable the exits. The following is an SQL example:

```
update schema_name.ICMRMControl
set PreStorExitPgm = 1
```

You should use the schema name of your Resource Manager tables in the SQL. Note, because there is one Resource Manager database per Resource Manager instance, there is only one row in the ICMRMCONTROL table similar to that of the ICMSTSYSCONTROL table for Library Server.

You must make the exit programs available to the Resource Manager before activating the exits through the flags in the ICMRMCONTROL table.

### 5.8.3 Making exit programs available to Resource Manager

You need to make the exit program available to the HTTP server that runs your Resource Manager. You do this by either adding a new dedicated load library to the STEPLIB list of your HTTP procedure, or by copying the exit modules into an existing STEPLIB used by the HTTP server.

### 5.8.4 What can you do in Resource Manager exits

You can use the Resource Manager exit program to override the default collection assigned by the Library Server when storing an item in OAM.

Depending on your configuration, the Library Server uses the collection assigned to the item type or the user when sending the store request to Resource Manager. The limitation is that you can only define one collection, which by default, is used for all the store requests for an item type by the user.

You may need to implement a more advanced approach to select other collections when storing data. For example:

- ▶ You may wish to change the collection name based on the date so you can use a different collection per quarter or month.



- ▶ You may want to use a group of collections where the store requests loop through the list for each subsequent store request to spread the load over a group of collections.
- ▶ You may choose to control the collection based on the object size of the document being stored.

One of the challenges in developing an exit is that you must implement the logic in your exit based on the limited information available to the exit. There is no information on the document key fields or item type that is passed to the exit.

If you want to implement advanced logic, another challenge is to distinguish between the requests.

One approach is passing additional information to allow the exit to perform logic depending on the object type or other fields in the structure passed into it. You can assign different collections based on this information.

**Important:** When implementing a Resource Manager exit, remember that the same exit is called for *all* similar requests from *all* users for *all* item types.

### 5.8.5 Illustrating the use of the PRE- and POSTSTORE exit

In Figure 5-4, we show the steps involved in storing an object:

1. The user requests the Library Server to store a document for a given item type.
2. The Library Server looks up (in the item type definition table) which Resource Manager to route the store request to and what the default collection for the item type is.
3. The component is registered with the default information. The resource table and the store request are passed to Resource Manager.
4. Resource Manager receives the request. Before calling OAM to store the object, PRESTORE exit is invoked.
5. The PRESTORE exit examines the passed information and updates the collection name if necessary.
6. The updated collection name is returned to Resource Manager and the object is stored in OAM based on the updated collection name.

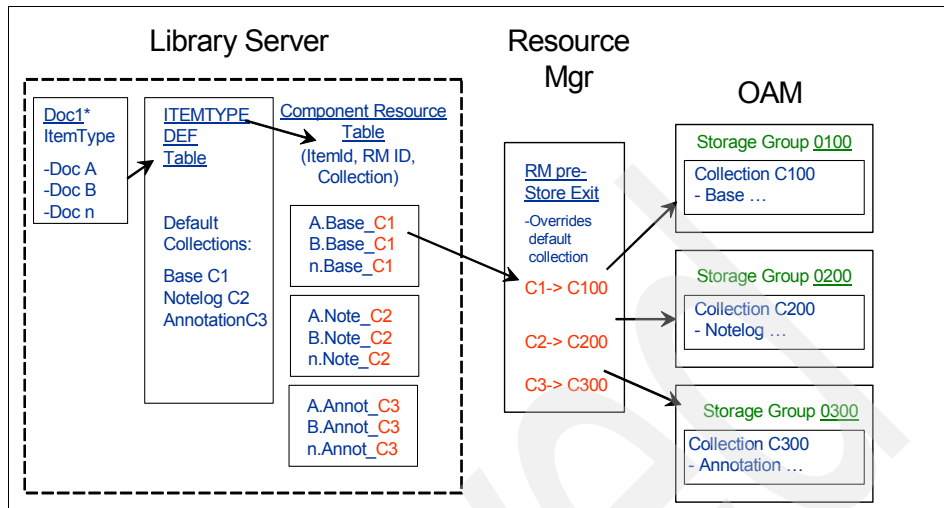


Figure 5-4 Processing store request and override default collection as required

After completing the store request in OAM, the request is processed as illustrated in Figure 5-5:

1. The POSTSTORE exit is called by Resource Manager with the updated information about the object.
2. Resource Manager returns a reply to the Library Server about the completed store request.
3. Information about the object is updated in the resource table for the component based on the information returned by Resource Manager. The Library Server uses this information such as the collection name for subsequent retrieve requests to Resource Manager.

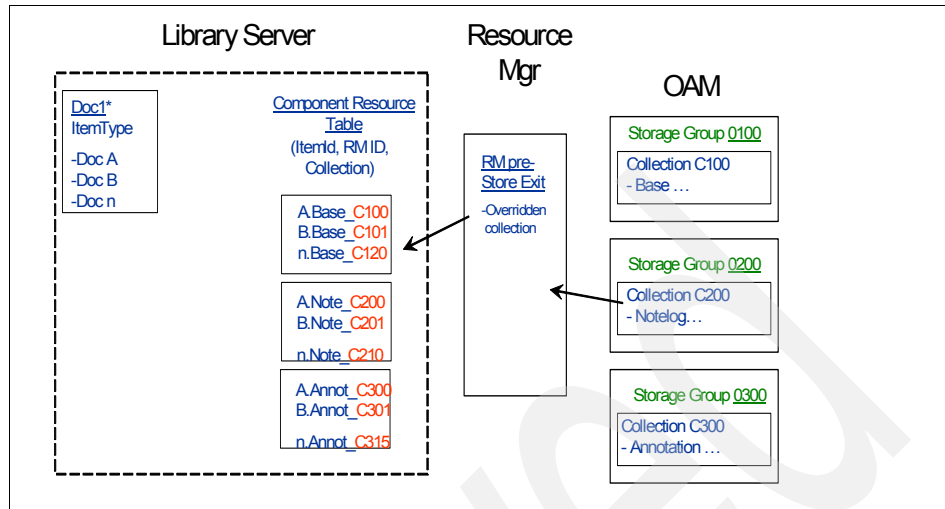


Figure 5-5 Returning updated collection information to Library Server





## Part 2

# Installation and migration

In this part, we describe Content Manager for z/OS V8.3 installation. In addition, we cover the migration to Content Manager for z/OS V8.3 from the following products:

- ▶ ImagePlus OS/390
- ▶ Content Manager V2.3 for OS/390 (also known as V7.1)
- ▶ Content Manager V7 for Multiplatforms



# Installation

In this chapter, we describe the process of installing IBM DB2 Content Manager for z/OS V8.3.

We cover the following topics:

- ▶ Content Manager installation and configuration
- ▶ Content Manager Toolkit for z/OS V8.3 installation
- ▶ System administration client installation
- ▶ Content Manager installation verification

## 6.1 Content Manager installation and configuration

In this section, we cover IBM DB2 Content Manager for z/OS installation and configuration.

Specifically, we discuss how to install and configure the following components:

- ▶ Library Server
- ▶ Resource Manager

Note, the SMP/E installation procedure is not covered in this IBM Redbook. For information regarding SMP/E installation, please refer to:

- ▶ *Program Directory for the Library Server for z/OS V08.03.00*
- ▶ *Program Directory for the Resource Manager for z/OS V08.03.00*

These documents come with the product tapes.

### 6.1.1 Planning for installation

Before you begin the installation, make sure that all the software and hardware requirements are satisfied.

Installation planning also includes completing the following tasks:

1. Estimate space for the Library Server and Resource Manager DB2 tables.
2. Select values to substitute in the installation jobs.
3. Define at least one Workload Manager (WLM) application environment for the Library Server.
4. Ensure DB2 stored procedure address space is operational.
5. Ensure the HTTP server is operational.
6. Take care of code page considerations.

If you need to offer a language character set other than the default DB2 Content Manager code page, which is US English, there are code page considerations. If you use other code pages, such as UK English, some information can be misread by the Library Server. If you intend to use another code page, we recommend using EBCDIC 0037.

7. Ensure you have a TSO user ID that can run DB2 jobs. It must have authority to:
  - Create databases, table spaces, and tables.
  - Bind application plans.
  - Grant run authority on application plans.
  - Grant all authority on tables.



8. Define three TSO user IDs for the customization.
9. Customize your storage medium.

For information about these tasks, please refer to the following manual:

- *IBM DB2 Content Manager for z/OS: Planning and Installing Your Content Management System*, GC18-7698

We recommend you use this chapter as complementary documentation to the planning and installation manual.

## Defining a WLM application environment

To define a WLM application environment, you need to do the following:

1. In the WLM menu, select **option 9** (Application Environments).
2. Under the Application Environment menu, select **action 1** (Create).  
Figure 6-1 appears.

Application-Environment		Notes	Options	Help
-----				
Create an Application Environment				
Command ==> _____				
Application Environment	...	ICMWLM01		Required
Description	...	WLM DEFINITION FOR CM		
Subsystem Type	...	DB2	Required	
Procedure Name	...	ICMWLM01		
Start Parameters	...	DB2SSN=DB2C,NUMTCB=10,APPLENV=CMWLM01		
-----				
Limit on starting server address spaces for a subsystem instance:				
1	1.	No limit		
	2.	Single address space per system		
	3.	Single address space per sysplex		

Figure 6-1 Creating an application environment

The window contains the following fields:

- Application Environment: Specifies the name of the application environment. It may be up to 18 characters.
- Description: Describes the application environment. It is an optional field.
- Subsystem Type: Defines the subsystem type that is permitted to use the application environment. *It must be DB2.*

- Procedure Name: Defines the JCL procedure that WLM uses to start server address spaces for the application environment. Each procedure must have its own WLM application environment. It does not need to be the same name as the WLM application environment.
  - Start Parameters: Specifies the parameters that WLM uses to start the JCL procedure.
  - Limit on starting server address spaces for a subsystem instance: Limits the number of address spaces to be created.
3. After entering the appropriate values for the fields, save this window, and go back to the Definition Menu.
  4. In the Utilities menu, choose **option 6** (Validate Definition). See Figure 6-2. Ensure the resulting message is as follows:  
  
IWMA045 No errors were found during validation of the service definition.
  5. In the Utilities menu, choose **option 1** (Install Definition). See Figure 6-2.

Figure 6-2 Install definition

8. To make sure that the WLM application environment is set, issue the following MVS™ command from the console or SDSF log:

```
/D WLM,APPLENV=*
```

It should show a message similar to the following:

```
IWM029I  19.14.06  WLM DISPLAY 981
          APPLICATION ENVIRONMENT NAME      STATE      STATE DATA
          ICMWLM01                          AVAILABLE
```

## 6.1.2 Library Server installation

The jobs related to the Library Server installation can be found in ?ICM?.SICMINS1.

We recommend copying these jobs from ?ICM?.SICMINS1 to another data set and customizing the jobs in the new data set.

After the Library Server installation, you need to perform the following tasks:

There are four steps involved in the Library Server installation:

1. Preallocate data sets.
2. Create DB2 tables and load data.
3. Create user tables.
4. Customize and copy the Workload Manager procedure.

### Step 1: Preallocate data sets

This step is needed if ICMMLSQ2 or ICMMLSQ3 is used.

You need to allocate the following data sets:

```
*.PROCLIB
*.ICMUSQL
```

**Note:** Since Content Manager for z/OS V8.3 no longer generates the access modules, you do not need to create the following data sets:

```
*.ICMULOAD
*.ICMUDBR
*.ICMUPGM
```

### Step 2: Create DB2 tables and load data

Customize and run the following jobs in the order presented here:

1. ICMMLSCR

This job creates DB2 objects (database, table spaces, tables, stored procedures, and UDF).

**Note:** The stored procedures and UDFs use the WLM application environment you created previously.

## 2. ICMMLSLD

Loads initial data into the Library Server database.

**Note:** This job loads a dummy password for the Resource Manager user ID into CM8 table ICMSTResourceMgr. After you complete the installation, use the system administration client to update the Resource Manager password.

## 3. ICMMLSBD and ICMMBIND

Performs the bind for the Library Server packages and plan. ICMMLSBD is the JCL and ICMMBIND contains the bind statements.

## 4. ICMMLSGT

Performs the grant for the plans, packages, procedures, and functions.

## 5. ICMMCACL

Initializes the compiled ACL table with the user IDs and privileges.

**Tip:** If for some reason you need to drop the Library Server database and recreate everything again, do not issue DB2 SQL **drop database** because it does not delete the stored procedures and UDFs. Run ICMMLSDR instead because the job is designed for this purpose.

## Step 3: Create user tables

When an item type is created, Content Manager creates a DB2 table corresponding to that item type.

Content Manager can create the tables by running jobs or dynamically creating them. This depends on whether the DD card //ICMSQL is specified in the WLM JCL procedure for the Library Server.

If ICMSQL DD card is specified, it runs one of the following SQL jobs which you will need to customize:

### ► ICMMLSQ1

Content Manager will insert the SQL statement and will submit the job automatically. It will not save the customized job as a PDS member.

Use this member only if you are not interested in modifying or saving the JCL/DDDL used to create the item-related tables.

► ICMMLSQ2

Content Manager will insert the SQL statement and will save the job as a PDS member. It will not execute the job automatically. You can configure the generated job to conform with the environment and run it later.

Use this member if you want to review DDL before submitting them or using multiple DB2 table spaces. Note, if the DB2 table space is not modified in each job, all the new tables will be created in the same table space.

► ICMMLSQ3

Content Manager will insert the SQL statement and will submit the job automatically. Unlike ICMMLSQ1, the ICMMLSQ3 job will save the customized job as a PDS member.

Use this member if you are not interested in modifying the JCL/DDDL used to create the item-related tables prior to executing the job, and you want to keep the job source.

#### **Step 4: Customize and copy the workload manager procedure**

Customize the sample WLM procedure, ICMMLSWL. You need to inform the WLM application environment previously created which job from Step 3 is to be used if //ICMSQL is specified.

After the customization, copy the procedure to a system procedure library.

#### **Post installation tasks**

After the Library Server installation, you need to run the following commands from a system administration client workstation and respond to the prompts as appropriate:

```
Installation drive \icmroot\bin\cmbenv81
```

```
Installation drive \icmroot\config\cmcfgls -t comtypes -l logfile
```

```
Installation drive \icmroot\config\cmcfgls -t predefs -l logfile
```

The last command creates sample attributes and item types so that the installation can be validated; it provides sample entities with which you can validate the installation.

If you do not run the above commands, you will get the following error when running the Client workstation to retrieve or import a document:

DGL3608A: DLL not ready: ICM7007: The access module required to access a component table has not been built correctly. The server log contains the name of the access module and the component type that must be built. Delete and

re-create the item type and verify the access module is correctly built.  
[STAT]:[LS RC=7007]

For more information about the step above, refer to 6.3, “System administration client installation” on page 169.

### 6.1.3 Resource Manager installation

The jobs related to Resource Manager installation can be found in ?ICM?.SICMINS2.

We recommend copying these jobs from ?ICM?.SICMINS1 to another data set and customizing the jobs in the new data set.

There are two steps involved in Resource Manager installation:

1. Create DB2 tables and load data.
2. Configure the HTTP server.

#### Step 1: Create DB2 tables and load data

Customize and run the following jobs in the order that they appear here:

1. ICMMRMCR

This job creates the DB2 objects to Resource Manager.

2. ICMMRMLD

This job loads the Resource Manager definitions into the DB2 tables.

3. ICMMRMBD

This job binds the Resource Manager packages and plan.

4. ICMMRMGT

This job grants execution privileges for Resource Manager plans.

**Note:** If for some reason, you need to drop the Resource Manager database and recreate everything, the ICMMRMDR job is provided to drop the database.

#### Step 2: Configure the HTTP server

Customize the sample JCL HTTP STC procedure, ICMMRMWB. Copy it to a system procedure library. You can create any name for the procedure name.

Insert the following directive in your httpd.conf configuration file:

Service /ICMResourceManager\* /pathname/icmmosct.so:myservice

**Note:** The sample procedure ICMMRMWB points to the HTTP *configuration file* /etc/icmmrmcf.conf. To create this icmmrmcf.conf, you can make a copy of the sample /etc/httpd.conf file, customize it, and rename it to this file.

You can also modify the procedure to point to another configuration file.

Once the httpd.conf configuration file is modified, stop and restart the HTTP server.

To start the HTTP server, issue the following MVS command from the console or SDSF log:

```
/S ICMMRMWB
```

To stop HTTP server, issue the following MVS command from the console or SDSF log:

```
/P ICMMRMWB
```

#### 6.1.4 Post installation

In our testing scenario, we configured the Library Server and Resource Manager in the same system.

To administer the Content Manager, you can use the system administration client. It runs in Windows, AIX, Solaris™, or Linux. We cover the system administration client installation in a Windows environment in 6.3, “System administration client installation” on page 169.

## 6.2 Content Manager Toolkit for z/OS V8.3 installation

The Content Manager Toolkit is a set of connectors available in z/OS UNIX System Services. It is designed to enable data access to Content Manager from applications that run in z/OS.

Note, we do not cover SMP/E installation in this IBM Redbook. For information about the SMP/E installation, please refer to the following document, which is shipped with the install tape:

► *Program Directory for IBM DB2 Content Manager Toolkit for z/OS V8.3*

Make sure that the prerequisites are satisfied because they are not the same as the prerequisites for the Content Manager server.

In our test scenario, we use the type 4 driver (remote client). The type 4 driver is implemented to DB2 as JCC driver, and it provides a new and completely separate JDBC driver implementation.

Type 2 connection is also supported.

Before running the customization jobs, the user who runs the jobs needs a .profile created and configured in z/OS UNIX System Services with the following environment variables:

► **PATH**

It should contain java/bin directories and the JCC directories.

► **STEPLIB**

It should contain DB2 libraries SDSNEXIT, SDSNLOAD, and SDSNLOAD2.

Note, the STEPLIB setting is for type 2 connection. For type 4 connection, you do not need the STEPLIB setting.

► **CLASSPATH**

It should contain the JCC sqlj.zip, db2jcc.jar, db2jcc\_license\_cisuz.jar, and db2jcc\_javax.jar.

**Important:** The legacy JDBC classes should not be in the CLASSPATH because there are some classes with the same name in JCC and legacy JDBC. Depending on the order it is inserted in the CLASSPATH, it may override some JCC classes and cause unpredictable results.

In our test scenario, because we have the legacy JDBC classes in the CLASSPATH, we got an error when running the sample SConnectDisconnectICM as shown in Example 6-1.

*Example 6-1 Error message when JDBC classes are in the CLASSPATH*

---

```
Exception in thread "main" java.lang.NoSuchMethodError:
com.ibm.db2.jcc.DB2BaseD
ataSource: method parseInt(Ljava/lang/String;I)I not found
    at com.ibm.db2.jcc.b.pb.a(pb.java:206)
    at com.ibm.db2.jcc.b.n.h(n.java:258)
    at com.ibm.db2.jcc.b.n.<clinit>(n.java:246)
    at com.ibm.db2.jcc.DB2Driver.<clinit>(DB2Driver.java:39)
    at java.lang.Class.forName1(Native Method)
    at java.lang.Class.forName(Class.java:180)
    at
com.ibm.mm.sdk.internal.sql.db2.PDB2ConnectionICM.setDefaultDriver(PDB2Connecti
onICM.java:42)
    at com.ibm.mm.sdk.server.PDB2UtilICM.setJDBCDriver(PDB2UtilICM.java:77)
```



```
at
com.ibm.mm.sdk.server.DKDatastoreICM.setJDBCDriver(DKDatastoreICM.java:11568)
at
com.ibm.mm.sdk.server.DKDatastoreICM.connect(DKDatastoreICM.java:1185)
at SConnectDisconnectICM.main(SConnectDisconnectICM.java:225)
```

---

## 6.2.1 Configuring Toolkit

After the SMP/E installation, we must configure the connector.

We need to run the jobs located in SICMSAMP data set.

Even if the job finishes with MAXCC=0, it does not necessarily mean the job ran correctly. You need to examine the job output to make sure all steps finished with COND CODE = 0.

### ICMCONFG

The ICMCONFG job runs the configuration script, icmconfig, to create the configuration files:

- ▶ <prefix>/usr/lpp/icm/V8R3M0/cmgmt  
cmbcmenv.properties  
ibmcmconfig.properties
- ▶ <prefix>/usr/lpp/icm/V8R3M0/cmgmt/connectors  
cmbicmenv.ini  
cmbicmsrvs.ini  
cmbcs.ini
- ▶ <prefix>/usr/lpp/icm/V8R3M0/bin  
cmbenv81.sh

In Example 6-2, we show the tailoring parameters specified in the ICMCONFG job. Note, the line numbers are not part of the actual input. We put them here to explain these lines.

*Example 6-2 Tailoring parameters in job ICMCONFG*

---

1. //STDENV DD \*
2. ICM
3. IBMCMROOT = /usr/lpp/icm/V8R3M0
4. DATABASENAME = DBOB
5. SCHEMANAME = IFVTJ
6. LSID = ICMCONCT
7. LSPW = ????????
8. ENCODE = EBCDIC

The parameters shown in Example 6-2 are used to build the content of the INI files. Note the following:

- ▶ In line 3, you specify the fully qualified path into which product is installed.
- ▶ In line 4, you specify the location name of your DB2 subsystem. You can see this name in the DB2 started task DB1BMSTR.
- ▶ In line 5, you specify the schema name used for Library Server tables.
- ▶ In line 6 and 7, you specify the user ID and password for the Content Manager connect user.
- ▶ In line 8, you specify the encoding (that is the code page) in which the INI files are created. You must specify EBCDIC if you plan to use the Java samples from the z/OS UNIX System Services (USS) environment. If you specify ASCII, you can neither edit them using the ISPF editor on z/OS nor can they be read by the Java programs.

The script icmconfig also calls a Java program that encrypts the user ID and password and stores it in cmbicmenv.ini file.

This job also writes two output files into z/OS UNIX System Services:

- ▶ <prefix>/usr/lpp/icm/V8R3M0/configstdout: Contains the execution output.
- ▶ <prefix>/usr/lpp/icm/V8R3M0/configstderr: Contains the error output.

To run this job, do the following:

1. Ensure the TSO logon procedure has a large REGION Size; otherwise, you can get message JVMDG218, located in the error output file configstderr.

In our test scenario, we set REGION to 262144 in our TSO logon procedure.

2. Put a large REGION SIZE in the EXEC PGM step; otherwise, you get error messages regarding memory issues when running the script.

In our test scenario, we set REGION=0M in our job ICMCONFIG.

3. ICMCRYPT

This job adds additional user IDs and passwords to cmbicmenv.ini. It is not needed to run for initial configuration.

4. ICMUNTAR

This job untars samples.pax. It is a compacted file that contains the samples.

5. ICMVERIFY

This job verifies that the Toolkit can connect to Content Manager.

## 6.2.2 Configuring Content Manager Toolkit for DB2 type 4 connection

DB2 type 4 connection uses a remote connection to the DB2 subsystem over TCP/IP. This connection type can be used both to connect to a Library Server on the local DB2 subsystem as well as to a Library Server running on a remote DB2 subsystem.

In Example 6-3, we show the values needed in the `cmbicmsrvs.properties` file to use a DB2 type 4 connection (note, the leading line numbers are not part of the INI file).

*Example 6-3 Configure Content Manager Toolkit to use DB2 type 4 connection*

---

```
1. ICMSEVER=ICMNLSDDB
2. ICMSCHEMA=IFVTJ
3. ICMSEVERREPTYPE=DB2
4. ICMSSO=FALSE
5. ICMDBAUTH=SERVER
6. ICMREMOTE=TRUE
7. ICMHOSTNAME=9.30.128.227
8. ICMPORT=8060
9. ICMREMOTEDB=DB0B
10. ICMNODENAME=
11. ICMOSTYPE=
```

---

Note the following about the parameters:

- ▶ In line 1, you specify the name you use in your programs when connecting to Library Server. The name is independent of the values you used when defining the Library Server database. Note, it must match the value specified in the `cmbicmenv.ini` file where information about the connect user ID is saved.
- ▶ In line 2, you specify the schema name for the Library Server tables.
- ▶ In line 6, you select the DB2 type 4 connection by specifying `ICMREMOTE=TRUE`.
- ▶ In line 7 and 8, you specify the TCP/IP address (or alternatively the host name) where the remote DB2 subsystem runs and the port number to which to connect.
- ▶ In line 9, you specify the location name for the remote DB2 subsystem. This value can be determined by looking into the DB1BMSTR started procedure of DB2.

## 6.2.3 Configuring Content Manager Toolkit for DB2 type 2 connection

DB2 type 2 connection uses a local connection to the DB2 subsystem. This connection type can only be used to connect to a Library Server on the local DB2 subsystem.

In Example 6-4, we show the values in the cmbicmsrsvs.properties file to use a DB2 type 2 connection (the leading line numbers are not part of INI file).

*Example 6-4 Configure Content Manager Toolkit to use DB2 type 2 connection*

---

```
1.  ICMSEVER=DB0B
2.  ICMSCHEMA=IFVTJ
3.  ICMSEVERREPTYPE=DB2
4.  ICMSSO=FALSE
5.  ICMDBAUTH=SERVER
6.  ICMREMOTE=FALSE
7.  ICMHOSTNAME=
8.  ICMPORT=
9.  ICMREMOTEDB=
10. ICMNODENAME=
11. ICMOSTYPE=
```

---

Note the following about the parameters:

- ▶ In line 1, you specify the location name for the remote DB2 subsystem. This value can be determined by looking into the DB1BMSTR started procedure of DB2.
- ▶ In line 2, you specify the schema name for the Library Server tables.
- ▶ In line 6, you select the DB2 type 2 connection by specifying ICMREMOTE=FALSE.

## 6.2.4 Using the JDBC Universal Driver for DB2 type 2 connection

The DB2 Universal JDBC Driver is a new JDBC and SQLJ driver implementation that is provided on multiple DB2 platforms and that can execute as a locally connected driver using native DB2 interfaces (type 2), or as a standalone, remote pure Java client using DRDA® interfaces (type 4).

JDBC drivers of various types have previously been provided with DB2 on each of the supported hardware platforms. The DB2 Universal JDBC Driver consolidates all of those drivers into a single common implementation.

The DB2 Universal JDBC Driver is provided under the same FMID JDB7712, but it resides in a different sub-directory (for example, if the legacy driver is installed

under /usr/lpp/db2/db2710, then the new DB2 Universal Driver will be found under /usr/lpp/db2/db2710/jcc).

In some externalized names (such as a sub-directory), the acronym JCC may be seen. This represents an IBM-internal name for the driver meaning Java Common Connectivity.

You must configure the JDBC driver with regard to the DB2 subsystem to which it connects. At a minimum, you must specify the Subsystem ID (SSID) of the DB2 subsystem to which you wish to connect. This can either be done by:

- ▶ Passing run-time options to the Java Run-time environment.
- ▶ Passing the name of a properties file at run time to the Java Run-time environment.
- ▶ By using the default properties file DB2JccConfiguration.properties used by the JDBC driver.

By default, the JDBC driver will try to locate a DB2JccConfiguration.properties file to get the run-time options if you do not pass any parameters when executing the Java command. The standard installation of the JDBC driver includes a sample properties file that only contains comment lines.

You can tailor your DB2JccConfiguration.properties file to specify the SSID of your DB2 subsystem. Example 6-5 shows the sample content of our properties file.

*Example 6-5 SSID specified in the DB2JccConfiguration.properties file*

---

```
db2.jcc.ssid=DB0B
```

---

The JDBC driver uses the standard Java search order via the CLASSPATH to locate the properties file.

If you have not tailored the properties file for your local environment and you do not specify specific run-time options, the driver will default to use a SSID of DSN. This will cause a run-time error shown in Example 6-6 when running the toolkit samples.

*Example 6-6 Error received when using the wrong SSID with the JCC driver*

---

```
$ java SConnectDisconnectICM DB0B ICMLS password
=====
IBM DB2 Content Manager                v8.3
Sample Program: SConnectDisconnectICM
-----
Database: DB0B
UserName: ICMLS
```

```

=====
Connecting to datastore (Database 'DBOB', UserName 'ICMLS')...
*****FAILED in Connection.finalize and try to
ROLLBACK*****com.ibm.db2.jcc.t2zos.T2zosConn
ection@17c5a888
*****FAILED in
Connection.finalize*****com.ibm.db2.jcc.t2zos.T2zosConnection@17c5a888
*****FAILED in Connection.finalize and try to
ROLLBACK*****com.ibm.db2.jcc.t2zos.T2zosConn
ection@50666888
*****FAILED in
Connection.finalize*****com.ibm.db2.jcc.t2zos.T2zosConnection@50666888

XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
X    !!! Exception !!!    X
XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
      Name: DKDatastoreAccessError
      Message: DGL0394A: Error in ::DriverManager.getConnection;

[IBM/DB2] [T2zos/1] T2zosConnection.flowConnect:DB2AttachInterface:initRRSAFAttac
h:564:RRS
      Identify failed,Return Code=12, Reason Code=X00F30006,
      Subsystem ID:DSN ,Plan Name:,Pklist:NULLID.* (STATE) : ;
      [SERVER = DBOB, USERID = ICMCONCT, SQL RC = -99999, SQL STATE = ]
      Message ID: 394
      Error State: ; [SERVER = DBOB, USERID = ICMCONCT, SQL RC = -99999, SQL STATE =
      ]
      Error Code: -99999
      com.ibm.mm.sdk.common.DKDatastoreAccessError:
      DGL0394A: Error in ::DriverManager.getConnection;
      [IBM/DB2] [T2zos/1] T2zosConnection.flowConnect:DB2AttachInterface:initRRSAFAttac
      h:564:RRS
      Identify failed,Return Code=12, Reason Code=X00F30006,
      Subsystem ID:DSN ,Plan Name:,Pklist:NULLID.* (STATE) : ;
      [SERVER = DBOB, USERID = ICMCONCT, SQL RC = -99999, SQL STATE = ]
      at
      com.ibm.mm.sdk.server.DKDatastoreICM.connect(DKDatastoreICM.java:3100)
      at SConnectDisconnectICM.main(SConnectDisconnectICM.java:243)
$

```

---

Notice the lines containing “Subsystem ID:DSN”. This is a good indicator of a wrong or missing properties file being used.

In our scenario, we place the DB2JccConfiguration.properties file in the directory /usr/lpp/icm/V8R3M0/cmgmt/ where the Content Manager properties files are kept, and then we make sure that this directory is at the beginning of the classpath list.

## 6.2.5 Details on DB2 Universal JDBC Driver setup and tailoring

For a detailed description on how to set up and tailor the DB2 Universal JDBC Driver, refer to the README file supplied in the installation directory of the DB2 Universal JDBC Driver.

If you use the default installation options, you can find this file in the following directory: `/usr/lpp/db2/db2710/jcc`.

The README file describes both the options you can specify in the `DB2JccConfiguration.properties` file as well as additional options you can use as run-time parameters on the Java command.

The README file also contains a list of prerequisite PTFs you need on your DB2 subsystem to use the JDBC driver.

## 6.3 System administration client installation

The system administration client enables you to manage your Content Manager system. Some of the tasks you can do with the system administration client include:

- ▶ Define user access and control.
- ▶ Define Resource Managers and their associated collections to the Library Server.
- ▶ Control access to documents.
- ▶ Set up Content Manager data model.
- ▶ Set up document routing.

In this section, we discuss the system administration client installation.

It is not our intention to repeat the detailed steps covered in the installation manual:

- ▶ *IBM DB2 Content Manager for z/OS: Planning and Installing Your Content Management System*, CG18-7698

Our goal is to provide the general overview and concentrate only on certain aspects of the installation we encountered during our installation to which you may need to pay special attention or address.

We recommend that you use this section as complementary documentation to the installation manual.

Before you proceed, make sure you have the correct authorities and that your system meets all the software and hardware requirements. You can find this information in the installation manual.

### 6.3.1 Installation process

Run the installation program by performing the following steps:

1. Enter the installation path.
2. Choose custom installation.
3. If you choose typical installation, the installation program will install both the Content Manager Library Server and Resource Manager onto your workstation.
4. Select the system administration client option and the Information Center (see Figure 6-3).

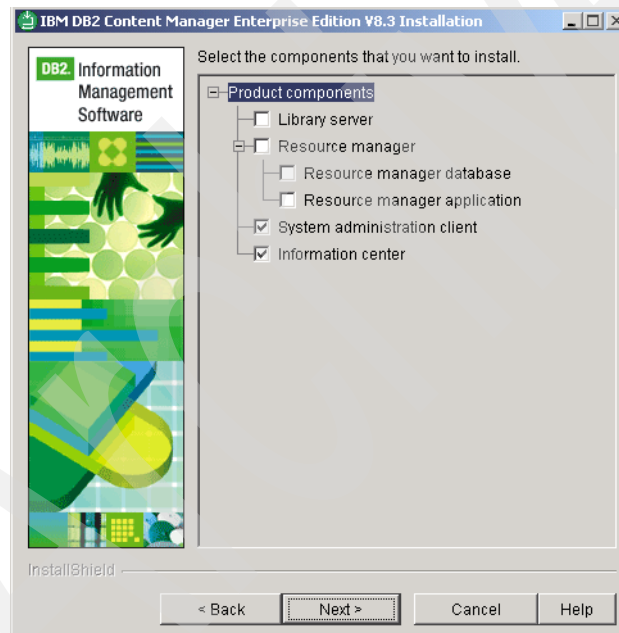


Figure 6-3 Content Manager installation: Custom install System Adm Client

5. Enter the working directory.
6. Set the configuration files location.  
In our installation, we set it to **Local**.
7. Set up the system administration client connection (see Figure 6-4).



Enter the fields with the DB2-related connection information. Although the Library Server is in a remote location, we do not have to check the option “Configure connector to remote Library Server database”. We are able to connect to the Library Server successfully.

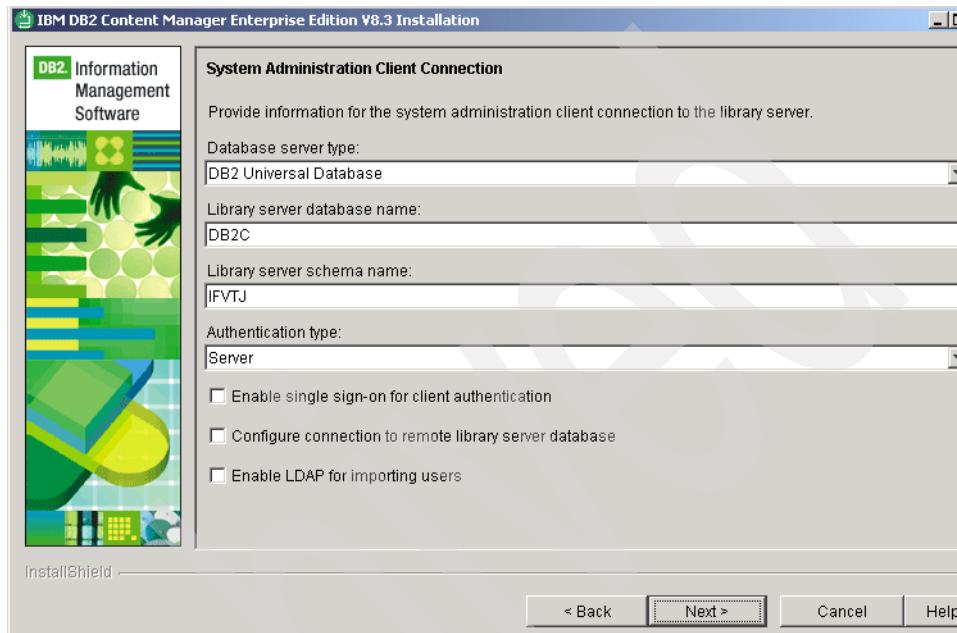


Figure 6-4 System administration client connection setup

8. Confirm the installation options, input values, and start the installation.

After the installation is finished, you need to catalog the DB2 z/OS database with the same alias name entered for the Library Server database name from the system administration client installation screen.

For information about how to catalog the DB2 z/OS database, refer to the installation manual:

- *IBM DB2 Content Manager for z/OS: Planning and Installing Your Content Management System, CG18-7698*

Chapter 6, “System administration client installation and configuration”, which is the section on cataloging remote node and database on z/OS.

The procedure on how to catalog a remote database described in the installation manual requires you to have at least a DB2 Connect. As DB2 Runtime Client does not have the functionality to connect directly to a DB2 z/OS, you need a

machine that contains DB2 Connect to work as a gateway between the workstation that has DB2 Runtime Client and DB2 z/OS.

If you installed the system administration client on a machine that contains only the DB2 Runtime Client, you need first to catalog the DB2 z/OS on a machine that contains DB2 Connect as described in the installation manual. After that, you need to catalog, in your DB2 Runtime Client machine, the DB2 Connect machine information. In your DB2 Runtime Client, you need to execute the following commands using a DB2 command window:

```
db2 catalog tcpip node <node_name> remote <hostname> server <port_number>
db2 catalog database <db_name> at node <node_name>
db2 terminate
```

where:

- ▶ <node\_name> is a name to identify the catalog.
- ▶ <hostname> is the host name or IP address of the DB2 Connect machine.
- ▶ <port\_number> is the DB2 Connect machine port number used to accept connections.
- ▶ <db\_name> is the database name that is cataloged in the DB2 Connect machine.

To make sure the connection is set correctly, test the connection by issuing the following command:

```
db2 connect to <db_name> user <userid> using <password>
```

It should return something similar to the following:

Database Connection Information

```
Database server      = DB2 OS/390 7.1.1
SQL authorization ID = IFVTX
Local database alias = DB2X
```

**Note:** The DB2 catalog procedure also applies to the Content Manager Client.

We are now able to use the system administration client. To log into the Library Server, we need to use the user ID entered in the installation job, ICMMLSLD.

After you log in, you notice there are no default item types, such as NOINDEX, created for that Content Manager server.

To create these item types, we need to run the commands from the system administration client workstation:

1. Execute the following command:

<installation drive \icmroot>\bin\cmbenv81

This sets the Content Manager environment variables. cmbenv81 is a batch file that needs to be configured before running it.

In our testing scenario, we changed the variables DB2PATH, removed the DB2 JDBC classes from CLASSPATH, and included the DB2 JCC classes as follows:

```
%DB2HOME%\JAVA\DB2JCC.JAR
%DB2HOME%\JAVA\DB2JCC_LICENSE_CU.JAR
%DB2HOME%\JAVA\DB2JCC_LICENSE_CISUZ.JAR
%DB2HOME%\JAVA\SQLJ.ZIP
```

2. Execute the following command:

<Installation drive \icmroot>\config\cmcfgls -t comtypes -l logfile

The system prompts you for connection information. Figure 6-5 records the prompts and our answers in bold. Use it as reference when you run the command.

```
Enter the RDBMS type 0-DB2 1-Oracle [Default: 0-DB2]
0
Enter the path to the directory where the Library Server is installed
[Default: C:\Program Files\IBM\db2cmv8]:
C:\Program Files\IBM\db2cmv8
Enter the DB2 instance path
[Default: c:\Program Files\IBM\sqllib]:
c:\Program Files\IBM\sqllib
Enter the database name (the default is "ICMNLSDB"):
DB2C
Enter the database administrator ID for the Library Server database
(the default is "ICMADMIN"):
IFVTJ
Enter the password of the database administrator ID
(the default is "password"):
*****
Enter the schema name for the Library Server database
(the default is "ICMADMIN"):
IFVTJ
The following are input parameters specified:
Library Server ID: 1
Database name: DB2C
Library Server database administrator ID: IFVTJ
Schema name: IFVTJ
Please review the input parameters specified above
(enter 1 to continue, enter anything else to start over again):
```

Figure 6-5 Configuration screen shot

In our test scenario, the log file containing the output of the cmcfgls execution is shown in Figure 6-6. The log file logs a list of component types as they are being built when the program executes.

```
2004-10-06 08:48:21 cmcfgls: Running LS configuration for RDBMS: DB2
2004-10-06 08:48:21 cmcfgls: Logging on to Library Server DB2C
2004-10-06 08:48:26 cmcfgls: Getting datastore definition
2004-10-06 08:48:27 cmcfgls: Listing components
2004-10-06 08:48:27 cmcfgls: Building component type: 200
2004-10-06 08:48:28 cmcfgls: Building component type: 201
2004-10-06 08:48:28 cmcfgls: Building component type: 209
2004-10-06 08:48:28 cmcfgls: Building component type: 202
2004-10-06 08:48:29 cmcfgls: Building component type: 203
2004-10-06 08:48:29 cmcfgls: Building component type: 204
2004-10-06 08:48:29 cmcfgls: Building component type: 205
2004-10-06 08:48:29 cmcfgls: Building component type: 208
2004-10-06 08:48:30 cmcfgls: Building component type: 210
2004-10-06 08:48:30 cmcfgls: Building component type: 206
2004-10-06 08:48:30 cmcfgls: Building component type: 207
2004-10-06 08:48:30 cmcfgls: Building component type: 300
2004-10-06 08:48:31 cmcfgls: Building component type: 301
2004-10-06 08:48:31 cmcfgls: Building component type: 302
2004-10-06 08:48:31 cmcfgls: Building component type: 303
2004-10-06 08:48:31 cmcfgls: Building component type: 304
2004-10-06 08:48:32 cmcfgls: Building component type: 400
2004-10-06 08:48:33 cmcfgls: Configuration of server database DB2C
completed successful
```

*Figure 6-6 Log file containing the output of running cmcfgls -t comptypes*

3. Run the following command:

```
<Installation drive \icmroot>\config\cmcfgls -t predefs -l logfile
```

The program asks with the same information as when we run it with **cmcfgls -t comptypes**.

In our test scenario, the log file containing the output of the command is shown in Figure 6-7. The log file logs a list of default item types as they are being created.

```
2004-10-06 08:49:32 cmcfls: Running LS configuration for RDBMS: DB2
2004-10-06 08:49:32 cmcfls: Generating predefined item types
2004-10-06 08:49:35 cmcfls: Creating item type NOINDEX
2004-10-06 08:49:43 cmcfls: Creating item type ICMSAVEDSEARCH
2004-10-06 08:49:50 cmcfls: Creating item type ICMFORMS
2004-10-06 08:49:54 cmcfls: Creating item type DOCROUTINGITEM
2004-10-06 08:50:00 cmcfls: Configuration of server database DB2C
completed successful
```

Figure 6-7 Log file containing the output of running `cmcfls -t predefs`

**Note:** Depending on how you set your `//ICMSQL` in your Library Server procedure, you need to manually submit the jobs generated in the PDS data set (ICMMLSQ2).

4. Log into your system administration client. Confirm that the item types have been created.

## 6.4 Content Manager installation verification

To verify that Content Manager for z/OS is installed successfully, we need to use Content Manager Client for Windows.

Install Client for Windows, log on, and import some documents in the NOINDEX item type.

After a successful import, search for the documents that you imported and retrieve them. If you are able to retrieve and view the documents, the installation process for Content Manager is successful.

### 6.4.1 Troubleshooting

There may be a variety of reasons if your installation was not successful. We address a few that we encountered during our testing.

#### Authentication issue

After we imported a document, we were not able to retrieve the document. We got an error message in the HTTP procedure as shown in Figure 6-8.

```
Authenticate.. no authorization data found, authenticate failed
Authentic User... -nobody-
Denied..... Authentic User required.
AA..... check returned 401
Translated.. "-null-"
HTLOADERRSRV. num=401 HTReason=3 log401error=1
ErrorLog.... Ý26/Oct/2004:21:16:06 +0400" /ICMResourceManager
return_code=401; HTTP_RESPONSE=401; HTErrorInfo=3; ERRORINFO=3
DetailedErr. bailing out on error 3 because No error page defined.
Returning... ERROR 401:
** IMW0216E Not authorized. Authentication failed.:
```

*Figure 6-8 Error message encountered*

This failure is because there is a limitation in the length of the URL that the z/OS HTTP server will accept. If you encode the parameter in the URL, the URL might become too long and the user ID and password will be truncated.

To prevent the "Not authorized" error, you need to set DisableURLCode=yes in c:\winnt\ICMCLIENT.ini file as follows:

```
[Options]
DisableURLCode=yes
[Scan]
PagesInFirstFile=5
PagesInRemainingFiles=15
```

### **Invalid Library Server and DB2 table definition**

It is also possible that an installation job does not complete successfully because your Library Server definition and the DB2 table definition do not match. To see if this is the case, refer to "Validate Library Server and DB2 table definitions" on page 387 and "Validate Library Server and DB2 view definitions" on page 390.

To solve this problem, you may have to manually rerun the jobs to ensure all of them execute successfully.

# Migrating from ImagePlus OS/390

In this chapter, we provide information about the migration from IBM Content Manager ImagePlus OS/390 to IBM DB2 Content Manager for z/OS V8.3.

We cover the following topics:

- ▶ Terminology
- ▶ Considerations and planning aids
- ▶ Prerequisites
- ▶ Assessing requirements and defining migration approach
- ▶ Planning and preparing for migration
- ▶ Installing Content Manager on z/OS
- ▶ Performing migration
- ▶ Performing post migration activities
- ▶ Customization alternatives

## 7.1 Introduction

If you currently use the IBM Content Manager ImagePlus for OS/390, commonly known as just ImagePlus, to manage your online document repository and are considering migrating to IBM DB2 Content Manager, we show you how to do so in this chapter through the experience we gained in completing this project.

Note, what we show you in this chapter is simply a guide to illustrate one of many possibilities. In certain places, we refer to excerpts from other publications.

We recommend you use the following publications for your primary sources of information:

- ▶ *IBM DB2 Content Manager for z/OS: Planning and Installing Your Content Management System*, GC18-7698
- ▶ *IBM DB2 Content Manager for z/OS: Migrating to Content Manager Version 8 for z/OS*, GC18-7699
- ▶ *IBM Object Distribution Manager MVS/ESA: System Programmer's Guide*, SC34-3124
- ▶ *IBM Folder Application Facility MVS/ESA Application Programming Interface: System Programmer's Guide*, SC34-3120
- ▶ *IBM Folder Application Facility MVS/ESA Folder and Workflow Application: System Programmer's Guide*, SC34-3117

### 7.1.1 Terminology

We assume you have a working knowledge of ImagePlus. When moving from ImagePlus to Content Manager for z/OS V8.3, one critical thing is understanding the terminology involved and how it may differ from what you are accustomed to using. Understanding new terminology and relating it to what you already know is frustrating. In an effort to minimize frustration and make the process easier, new and changed terms are defined in Table 7-1. Included in the list are abbreviations and what they mean as they are used in this chapter.

Table 7-1 New and old terms

Term	Definition
IPFAF/API	IPFAF Application Programming Interface
IPFAF/FWA	IPFAF Folder Workflow Application
IPFAF/WRAPI	IPFAF Workstation Remote API
IODM	ImagePlus Object Distribution Manager



Term	Definition
IP/390	IBM Content Manager ImagePlus for OS/390
LOB	Line of business application: A non-image application that takes advantage of the APIs provided to display, manipulate, or delete documents stored within an imaging repository, such as ImagePlus.
Multiplatforms	Operating system other than z/OS, such as Windows and AIX.
Library Server	Content Manager component that takes the place of IPFAF, IPFAF/API, and IPFAF/FWA. Note that Content Manager APIs may be either z/OS or multiplatforms, and in some cases both.
OAM	Object Access Method: DFSMS component on z/OS normally used for object storage and retention management.
Resource Manager	Content Manager component that provides the same function as IODM: interface with OAM.
TSM	Tivoli Storage Manager: Historically, the multiplatform support for document retention management. In Content Manager for z/OS V8.3, support is provided for TSM in addition to OAM.

### 7.1.2 Considerations and planning aids

A major consideration in determining the course and actions you take in performing a migration is why you want to migrate.

If your reason for migrating is purely to upgrade, and your implementation consists of a complete ImagePlus installation, and you do not intend to exploit any additional features or functionality in Content Manager, there are utilities that most likely meet your metadata migration needs.

The generic migration utilities provided by Content Manager for z/OS migrate the *metadata* from a complete ImagePlus system, which would be IODM, IPFAF/API, and IPFAF/FWA. If you have a custom front end application that is replacing IPFAF/FWA, or have an integrated Line of Business application with ImagePlus using the IPFAF APIs, additional custom work is necessary. We discuss this later in this chapter.

If your reasons for migrating are to take advantage of a more robust set of features and functions, and a much more flexible data model in Content Manager for z/OS, then your migration project requires more time and effort in planning and implementing the enhanced Content Manager solution.

To plan for your migration, the following checklist is provided as a starting point. We have also provided insight as to how you may take each item into consideration in making your decisions. This is not an exhaustive list, but it contains the basic considerations in assessing your migration needs. Note that these items are given in no particular sequence of importance, since what is important to one person may not be to another:

## Checklist

1. \_\_\_\_ Does your environment consists of the complete ImagePlus installation, IODM, IPFAF/API, and IPFAF/FWA?

*Considerations:* If you have a complete ImagePlus installation, the provided migration utilities may meet your need.

If you have replaced IPFAF/FWA with your own front end application, or you have a business partner solution, the migration utilities should serve as a starting point. You need customized utilities to complete your migration project.

2. \_\_\_\_ Do you have any Line of Business applications integrated with ImagePlus?

*Considerations:* You may have an application that interfaces with ImagePlus via IPFAF/API, IPFAF/WRAPI, and/or IODM APIs. In planning your migration, you need to consider how you will provide the same or similar interface after the migration.

An example is a claims processing system that is used to perform claims processing activities, such as the capability to determine eligibility and compute benefits. This application may be host-based, running in a CICS environment using the IPFAF/API, or a multiplatform application using IPFAF/WRAPI. The personnel processing the claims may not have access to the hard copy document, but rely on stored objects, such as scanned documents and other electronic data.

For users to have a seamless interface between the claims application and ImagePlus, the APIs provided would have been used. At a minimum, you need to rewrite the interface using the new set of APIs.

3. \_\_\_\_ Do you currently use IPFAF workflow, or are you considering implementing workflow after you migrate?

*Considerations:* The migration utilities do not migrate any workflow definitions or consider whether a document is currently in routing or not. To migrate

IPFAF workflow or the documents that are currently in workflow, you must develop a custom migration program.

Also note that a prerequisite for using the provided migration utilities is that the migrated documents cannot be in workflow.

If you want to implement workflow after you migrate, Content Manager provides workflow options. Working with Content Manager for z/OS, you can design your workflow process and implement it.

4. \_\_\_\_ How much DASD will you need?

*Considerations:* The Library Server database of a migrated IPFAF database requires, at a minimum, six times the DASD.

This figure does not include the DASD needed for the interim data sets used during the migration process. Two times the DASD needed for the resulting Library Server database is a good rule of thumb for interim/work data sets.

Note, Content Manager allows you to eliminate data fields that you may not use for some applications. This may reduce the DASD requirements.

5. \_\_\_\_ How many IPFAF applications, APPLIDCDs, do you have and why do you want them?

*Considerations:* Some users have each APPLIDCD in its own IPFAF tableset, which limits them to a maximum of eight applications, but it provides physical separation of the data. You should consider if you really need to have the data physically separated.

If you must have the data physically separated, you can accomplish this in Content Manager by using a different item type for each application.

If you consolidate applications/APPLIDCDs, you must take into account the possibility of duplicate data. For example, in APPLIDCD 01, you may have folder ID ABCDEFG. You may also have that same folder ID in APPLIDCD 02. You need to determine the relationship of the folders and take them into consideration. One possible solution may be to rename the duplicated data.

6. \_\_\_\_ How many IPFAF tablesets do you use and why?

*Considerations:* As indicated in the earlier discussion about APPLIDCDs, IPFAF tablesets provide physical separation of data. As with APPLIDCDs, if you consolidate the tablesets, you need to take into consideration that some duplicate entities may exist. One possible solution may be to rename the duplicated data.

7. \_\_\_\_ What is the most practical approach for your migration?

*Considerations:* There are many ways to migrate the ImagePlus data. The basic migration paths are as follows:

- Migrate all the data at one time: This is straightforward with no overlap.

- Incrementally: This provides an unlimited number of possibilities. For example, you can migrate one APPLICD at a time, or all folders and documents of a specific age at a time.
- Coexist with ImagePlus: You may want to run both the existing ImagePlus system and the new Content Manager for z/OS system in parallel. In this scenario, if users need to query results from both repositories at the same time, Content Manager offers the federated client.

8. \_\_\_\_ What are your client needs?

*Considerations:* A simple ImagePlus implementation consists of 3270 sessions for accessing the CICS or IMS IPFAF/FWA front end application and the IWPM clients for storing, viewing, and modifying documents.

Content Manager Information Integrator for Content provides a central access point for various content repositories which includes ImagePlus. For each supported repository, a connector, written to that repository's APIs, is needed to access, via eClient, a thin client. For ImagePlus, the connector is written to IPFAF/WRAPI.

Content Manager also has a federated client and a thick client.

The federated client provides simultaneous (federated) access to multiple repositories and presents the results in a consolidated list. Both the eClient and the federated client can be accessed via a browser.

The thick client, also known as pClient or winClient, must be installed on each user's workstation.

The eClient with ImagePlus has become commonplace in today's environment. With a migration to Content Manager, the same client can be used, just pointing to a different repository.

Depending on other requirements you may have, there may be a place for the thick or federated clients in your solution. You need to determine the requirements for your particular solution.

9. \_\_\_\_ What are your document capture needs?

*Considerations:* ImagePlus provides a wide variety of document capture alternatives.

There are several implementations of auto-indexing and batch processing available which can be fed by IWPM batch scanning, Business Partner solutions, and various custom applications.

Scanning is supported for users with the Windows client and an attached desktop scanner.

In addition, several Business Partner solutions allow users to implement document capture processes that incorporate various special requirements.

Content Manager provides document capture for users with the Windows client and an attached desktop scanner. All batch capture support is provided via Business Partner solutions. You need to contact the various Business Partners and understand their solutions to determine which solution best meets your needs.

10. \_\_\_\_ What are your print requirements?

*Considerations:* Printing is supported with all ImagePlus and Content Manager clients. ImagePlus also has batch print support provided by a Business Partner offering. Content Manager currently has no batch print support.

11. \_\_\_\_ Are the item type structure and attributes that the migration utility provides exactly what you want?

*Considerations:* You have some flexibility in specifying what attributes are loaded for the data that is being migrated. See 7.3, “Customization alternatives” on page 215 on how to accomplish this task.

12. \_\_\_\_ Are the names that are given to the attributes provided by the migration utility exactly what you want?

*Considerations:* Your options are very flexible for naming user-defined attributes. You can use names that make sense in your environment. See 7.3, “Customization alternatives” on page 215 on how to accomplish this task.

**Note:** The ImagePlus to Content Manager V8.3 migration utilities unload the ImagePlus Folder Application Facility tables, which are pure metadata. This data is then processed and reformatted to load into a Content Manager Library Server database.

The location of the target Library Server database can be in the same DB2 subsystem as ImagePlus or a completely separate LPAR. The major DB2 subsystem consideration for an ImagePlus migration is the location of OAM. The Content Manager V8.3 Resource Manager database must be in the same DB2 subsystem as OAM. Note the migration utilities only migrate the metadata and do not migrate the objects in OAM.

### 7.1.3 Prerequisites

The following references provide the prerequisite information:

- ▶ *Resource Manager Program Directories*
- ▶ *IBM DB2 Content Manager for z/OS: Planning and Installing Your Content Management System*, GC18-7698

Always consult the latest information available to determine what is supported and what is not. At the time of this redbook, the following ImagePlus product levels are required to utilize the migration utilities shipped with Content Manager for z/OS:

- ▶ ImagePlus Object Distribution Manager (IODM), V2.2 or V3.1
- ▶ ImagePlus Folder Application Facility/Application Programming Interface (IPFAF/API), V2.2 or V3.1
- ▶ ImagePlus Folder Application Facility/Folder Workflow Application (IPFAF/FWA), V2.2.1 or V3.1

**Important:** The migration utilities shipped with Content Manager for z/OS are designed for a complete ImagePlus for OS/390 implementation; therefore, *all* components are prerequisites. If any of these components is not part of your environment, the provided migration utilities cannot be used without modification.

## 7.2 Migrating ImagePlus for OS/390

In this section, we describe how we migrate an ImagePlus for OS/390 system to Content Manager for z/OS system. It is our intention for you to use the content as a general guide when you perform your ImagePlus system migration.

In our scenario, the migration consists of six primary steps:

1. Assess the requirements and define the migration approach.
2. Plan a migration strategy and prepare for the migration.
3. Install Content Manager for z/OS.
4. Perform the migration.
5. Perform post migration activities.
6. Test the migrated environment.

### 7.2.1 Assessing requirements and defining migration approach

Before you begin, you should assess your business requirements and define your migration approach.

#### **Our ImagePlus system environment**

In our scenario, our environment is relatively straightforward. We have a complete ImagePlus for OS/390 system, which consists of IODM V3.1, IPFAF/API V3.1, and IPFAF/FWA V3.1.

## Checklist input

In response to the checklist in 7.1.2, “Considerations and planning aids” on page 179, we have the following responses for our scenario:

1. \_\_\_\_ Does your environment consists of the complete ImagePlus installation, IODM, IPFAF/API, and IPFAF/FWA?

*Response:* Yes.

2. \_\_\_\_ Do you have any Line of Business applications integrated with ImagePlus?

*Response:* No

3. \_\_\_\_ Do you currently use IPFAF workflow, or are you considering implementing workflow after you migrate?

*Response:* No

4. \_\_\_\_ How much DASD will you need?

*Response:* Our ImagePlus database uses 83,486 cylinders on 3390 devices. Initially we planned for:

- Six times the ImagePlus database size, or 500,000 cylinders, for our interim processing data sets (work space)
- Three times the ImagePlus database size, or 250,000 cylinders, for our Content Manager Library Server database

Our migration process actually consumed the following DASD resources:

- 100,000 cylinders for the unloaded ImagePlus database and the OAM object directory table.
- 449,550 cylinders for the resulting output of the ICMMPx jobs, which create the data to be loaded in the Content Manager database:
  - 549,550 Cylinders were the total interim data set requirement, so we were relatively close with our estimate.
- 449,550 cylinders for our resulting Content Manager Library Server database, which were nearly double what we estimated.

**Important:** In our ImagePlus environment, we use secondary indexes on the IPFAF object name table, EYPTONAMxx. We do not use versioning.

Secondary indexes and usage of EYPTONAMxx would not require any additional space in the resulting Content Manager database. This is because the folder entry already has space for secondary indexes, and the object name in EYPTONAMxx would be used in place of the system-generated name.

If versioning is used, the interim data set and Content Manager database DASD would need to be increased. An estimate could be determined by comparing the number of version entries in IPFAF to the number of base objects and increase the estimated DASD by a proportionate amount.

5. \_\_\_\_ How many IPFAF applications, APPLIDCDs, do you have and why do you want them?

*Response:* We use only one APPLIDCD - 01.

6. \_\_\_\_ How many IPFAF tablesets do you use and why?

*Response:* We use only one tableset - 00.

7. \_\_\_\_ What is the most practical approach for your migration?

*Response:* Due to our relatively low volumes, we select a one time migration for everything.

8. \_\_\_\_ What are your client needs?

*Response:* All our users have casual scan requirements. We therefore select to use the thick client.

9. \_\_\_\_ What are your document capture needs?

*Response:* We use casual scanning on some workstations and batch scanning with the IPFAF Batch Store process for the majority of our input.

10. \_\_\_\_ What are your print requirements?

*Response:* We have no batch print requirements.

## Defining approach

After you assess your existing environment, you need to define your approach before you start.

Because we have a relatively simple environment with no special needs, there are several approaches for migrating our system:



- ▶ Migrate all data at once. This means there is only one migration pass where all data will be migrated to a single data model as provided in the migration utilities.
- ▶ Migrate incrementally. For example, you can migrate 12 months of data at a time, using receive date as the criteria. You can implement this by qualifying the select statements on the IPFAF unload jobs - ICMMMIFx.
- ▶ Migrate each form number to a separate item type in Content Manager.

We select to go with the first option: Migrate all at once. This is because our data volumes are relatively low and we have no unique requirements that would warrant separate item types.

## 7.2.2 Planning and preparing for migration

Planning for migration includes the following tasks:

- ▶ Ensure that there is adequate DASD.
- ▶ Determine the naming conventions and values for the JCL symbolics.

There are three sets of jobs in the SICMMIN1 data set for performing the migration:

- ▶ ICMMMIF1 - ICMMMIF9, and ICMMMIFA

This group of jobs unloads the applicable data from the ImagePlus DB2 tables, and the object names from the OAM storage group for identifying annotations. A last minute addition was the inclusion of the jobs to unload the object sizes from OAM to update the object size in Content Manager resource entries.

- ▶ ICMMMIP1 - ICMMMIP4

This group of jobs processes the data unloaded from the ImagePlus DB2 tables. The output of these jobs is data sets that will be loaded into Content Manager DB2 tables.

- ▶ ICMMMICT, ICMMMIC0 - ICMMMIC4, ICMMMIC6 - ICMMMIC9, and ICMMMICA

This group of jobs creates the additional Content Manager DB2 tables that will be needed for the migration, and loads the applicable Content Manager DB2 tables.

Each of the migration jobs contains symbolics that are data names enclosed in ?s. For example, the MSGCLASS parameter on the job statement is coded as:

MSGCLASS=?OUTC?

The symbolic value ?OUTC? needs to be replaced by the actual value. In our scenario O is to be used.

Before the migration jobs can be executed, all symbolic values must be resolved.

This part of the planning process may be the most time-consuming and will involve input from all areas associated with the project. System requirements and standards are some of the primary considerations for this process.

We grouped the symbolics by job function and assigned values for each group. Table 7-2 shows the symbolics and values we use for the ICMIIIIFx jobs, which unload the ImagePlus data from the DB2 tables.

*Table 7-2 ICMIIIIFx jobs: Used to unload the ImagePlus DB2 tables*

Symbolic	Our value	Description
?DSN?	SYS1.DB2L	high-level qualifier of DB2 system library - SDSNLOAD
?HLQ1?	ICMV830.IPMIG	high-level qualifier for the ImagePlus unload data sets. Each job could use different values, but we elected to use just one. Note also that this could represent more than one qualifier for a data set, which is how we are using it
?DB2SYS?	DB2L	DB2 subsystem
?DB2RUN?	DB2L.RUNLIB.LOAD	DB2 Runtime library
?CREATOR1?	ENTDB	DB2 creator ID for IPFAF/FWA tables
?CREATOR2?	EYPDB	DB2 creator ID for IPFAF/API tables
?CREATOR3?	EKCDB	DB2 creator ID for IODM tables
?APPLIDCD?	01	IPFAF application ID code
?XX?	00	IPFAF/API tableset used and the OAM storage group for unloading the OAM object directory table
?YY?	01	IPFAF application ID code
?OAMGRP?	GROUP53	OAM storage where IPFAF objects are stored for unloading object size

**Note:** The ?XX? symbolic is used for two purposes. In all jobs except ICMMMIF8, it represents the IPFAF/API tableset. In ICMMMIF8, it represents the OAM storage group that annotations are stored in. *Be careful!*

The migration utilities assume the OAM storage groups use GROUPxx names. If you use other names for the OAM storage groups, you need to replace the entire GROUP?XX? values with the group names you use.

Table 7-3 shows the symbolics and values we use for ICMMMIPx jobs, which process the data unloaded from the ImagePlus DB2 tables.

Table 7-3 ICMMMIPx jobs: Process unloaded ImagePlus data

Symbolic	Our value	Description
?HLQ1?	ICMV830.IPMIG	High-level qualifier for output data sets
?HLQ2?	ICMV830.SMPE	High-level qualifier for Content Manager Library Server SICMLMD1 data set
?HLQ3?	ICMV830.SMPE	High-level qualifier for Content Manager Library Server SICMMIN1 data set
?HLQ4?	ICMV830.IPMIG	High-level qualifier for ImagePlus unload data sets

Determining the values for the symbolics in the ICMMMICx jobs is more complex and requires additional considerations over what is needed for the ICMMMIFx and ICMMMIPx jobs.

Table 7-4 shows the symbolics and the associated values we use that are common throughout the ICMMMICx jobs. Additional tables define other symbolics.

Table 7-4 ICMMMICx jobs: Create additional CM entities and loads data

Symbolic	Our value	Description
?DB2LOAD?	SYS1.DB2L	high-level qualifier of DB2 system library - SDSNLOAD.
?DB2SYS?	DB2L	DB2 subsystem.
?DB2RUN?	DB2L.RUNLIB.L OAD	DB2 Runtime library.
?DB2PLAN?	DSNTIACM	DB2 Runtime plan name.
?BPS4?	BP0	DB2 buffer pool.

Symbolic	Our value	Description
?ICMDB?	ICMMLSDB	Content Manager Library Server DB2 database name.
?DATABASE?	ICMMLSDB	Content Manager Library Server DB2 database name. (Same value as ?ICMDB?)
?TBLSNAME?	TMPTBLS	DB2 temporary table space for processing object sizes.
?SQLID?	DB2LSADM	DB2 User ID with administration authority.
?OUTC?	O	MSGCLASS on JOB statement.
?CREATOR?	ICMADMM	Creator for additional DB2 database table.
?HLQ2?	ICMV830.IPMIG	High-level qualifier for the output data sets from the ICMMPx jobs that contain the data to be loaded into the Content Manager Library Server database. Note also that this could represent more than one qualifier for a data set, which is how we are using it.
?ICMADMIN?	ICMADM	Content Manager system administrator User ID that is defined to RACF and has DB2 system administration authority.
?RMPORTNUM	8092	Default Resource Manager TCP/IP port number.
?RMHTTPSPORTNUM?	8080	Default Resource Manager HTTPS TCP/IP port number.
DFLTRMCODE	2	Default Resource Manager code.
DFLTCOLLCODE	1	Default code for OAM collection.
DFLTPREFCHCOLLCODE	1	Default code for OAM pre-fetch collection.

The ICMRICT job defines the Content Manager Library Server component tables for the item types that will be required for loading the ImagePlus data.

The symbolics for the ICMRICT jobs require special consideration.

Content Manager assigns a unique component identifier for each new item type/component defined that is used to name the DB2 tables. This is normally handled automatically by Content Manager when the system administrator client is used to define new entities; however, when performing system migration using the migration jobs that create the required DB2 tables, we need to predetermine the values that will be needed and set them in the migration jobs.

The unique identifier (with ICMUT prefix) has two parts. The first part consists of five characters and it stands for component type ID. The second part consists of three characters and it stands for segment ID.

We use segment ID 001, which is standard. Your solution may use other segment IDs. For more information, refer to:

- ▶ *IBM DB2 Content Manager for z/OS: Planning and Installing Your Content Management System*, GC18-7698
- ▶ *IBM DB2 Content Manager for z/OS V8.3: System Administration Guide*, SC27-1335

The component IDs are assigned in sequence. Anything 01000 and above is user-defined, and everything below is reserved for system use. Note that when the Content Manager Library Server is initially installed, item types are defined; and although they are defined by the installation process, they are considered user-defined, and therefore, use ID values 01000 and higher.

All entities within Content Manager, such as attributes, components, and item types, have entries in the ICMSTNLSKEYWORD table with a unique identifier and description. The ICMSTMAXKEYWORD table is added in Content Manager V8.3 to keep track of the last value used for each entity. The type of identifier being tracked in this table is specified by the KEYWORDCLASS. For an ImagePlus migration, the following KEYWORDCLASSES are of interest to us:

- ▶ 1 - Attribute
- ▶ 2 - Item type
- ▶ 3 - Component type view
- ▶ 5 - Component type
- ▶ 18 - Item type view

To determine what has been used, query the ICMSTMAXKEYWORD DB2 table with the following SQL statement:

```
SELECT * FROM creator.ICMSTMAXKEYWORD
WHERE KEYWORDCLASS IN (1, 2, 3, 5, 18)
ORDER BY KEYWORDCLASS;
```

Figure 7-1 shows the results we get in our environment. To determine the first value we can use for each KEYWORDCLASS, we add 1 to each

MAXKEYWORD returned in the query. For example, to determine the first available value for component types, we look at the MAXKEYWORD value for KEYWORDCLASS 5, which we see is 1006. This makes 1007 and above available for us. Table 7-5 shows how we applied this information.

KEYWORDCLASS	LANGUAGECODE	MAXKEYWORD
-----+-----+-----		
1	ENU	1021
2	ENU	1003
3	ENU	1006
5	ENU	1006
18	ENU	1003

Figure 7-1 Results from ICMSTMAXKEYWORD DB2 query

**Note:** The ICMMIC1 job defines all attributes, components, and item types to Content Manager that are needed for an ImagePlus migration. One of the last actions that the job takes is to regenerate the ICMSTMAXKEYWORD table to reflect values used in the migration.

Table 7-5 shows the symbolics and our values.

Table 7-5 Component type IDs for Content Manager components

Symbolic	Our value	Description
?02000001?	01007001	Component ID for folder
?02001001?	01008001	Component ID for folder note
?02002001?	01009001	Component ID for document root
?02003001?	01010001	Component ID for document child
?02004001?	01011001	Component ID for document part

Rather than requiring that all users are limited to the same attributes as ImagePlus, Content Manager provides flexibility by defining item types for folders and documents with attributes that meet your specific business requirements.

The migration jobs that create the various item types that will be required for the ImagePlus data also define the necessary attributes. Each attribute has a unique attribute ID.

The attribute IDs are used in several ways. When a new item type is defined, the data field names in the Library Server Component DB2 table for the attributes are

named using the convention ATTRxxxxxxxx, where xxxxxxxxx is a unique 10 digit value.

In the ICMRICT migration job, which defines the required component tables, the data field name for the attributes has a symbolic value for the last five digits, which corresponds with the five digit attribute ID being used.

Note that although our migration, except for the symbolics, is using all the defaults provided, including item type attributes and attribute names, there is a significant amount of flexibility that you have here.

Two basic areas of flexibility include the ability to change the attribute name that is displayed for users to something more meaningful, and the ability to include only applicable attributes in a particular item type/component.

As with component type IDs, the next available value for an attribute ID needs to be determined, which is included in the query results in Figure 7-1 on page 192. The MAXKEYWORD for KEYWORDCLASS 1 gives us the last value used for attribute ID, which we see is 1021. This makes 1022 and above available for us.

Table 7-6 lists the symbolics for the attributes used in the ICMRICT job and the values we used.

*Table 7-6 Attribute IDs for component tables attribute data fields*

<b>Symbolic</b>	<b>Our value</b>	<b>Description</b>	<b>IPFAF field</b>
<b>?02000?</b>	01022	Folder ID	FOLDID
<b>?02001?</b>	01023	Application ID code	APPLIDCD
<b>?02002?</b>	01024	Folder token	FOLDTKN
<b>?02003?</b>	01025	Folder type code	FOLDTYCD
<b>?02004?</b>	01026	Empty folder indicator	EMPTFOLD
<b>?02005?</b>	01027	Code page	CODEPAGE
<b>?02006?</b>	01028	Security classification	SECURCL
<b>?02007?</b>	01029	Create date	CRTEDATE
<b>?02008?</b>	01030	Folder type	FOLDTYPE
<b>?02009?</b>	01031	Date changed	CHANGED
<b>?02010?</b>	01032	Folder description	FOLDDESC
<b>?02011?</b>	01033	User-defined data	USERDATA

Symbolic	Our value	Description	IPFAF field
?02012?	01034	Secondary index 1	SECINDX1
?02013?	01035	Secondary index 2	SECINDX2
?02014?	01036	Secondary index 3	SECINDX3
?02015?	01037	ImagePlus User ID	IPUSERID
?02016?	01038	Folder note short description	SHRTDESC
?02017?	01039	Folder note text	NOTETEXT
?02018?	01040	Time folder or document created	TIMECRTE
?02019?	01041	Date document received	RECVDATE
?02020?	01042	Date document filed	FILEDATE
?02021?	01043	Form number	FORMNUM
?02022?	01044	File tab	FILETAB
?02023?	01045	Object description	OBJDESC
?02024?	01046	User-defined date	USERDATE

The values we have not yet determined are the item type identifiers for the ICMIMIC1 job. As with component type identifiers, the item type identifiers are a unique number equal to or greater than 1000, and are also included in the query results in Figure 7-1 on page 192. The MAXKEYWORD for KEYWORDCLASS 2 gives us the last value used for item type ID, which we see is 1003. This makes 1004 available for us.

Table 7-7 shows the item type values we used.

*Table 7-7 Item type ID symbolics and values used*

Symbolic	Our value	Description
Item type IDs		
?IT2000?	1004	Folder item type ID
?IT2001?	1005	Document item type ID
Item type view IDs		
?ITV2000?	1004	Folder item type view ID
?ITV2001?	1005	Document item type view ID



Although the attribute IDs were used in the ICMMMICT job, and the component DB2 tables are defined, the attributes, item types, components, views, and keywords are actually defined to Content Manager in the job ICMMMIC1.

We have already determined the component type IDs, item type IDs, and attribute IDs we will use for the ICMMMICT job. We will use the same values for all subsequent jobs.

Although the same values will be used, the same symbolics are not always used in subsequent jobs, and the format is slightly different in some jobs. The primary difference is that the leading zero may not be used in all cases, or another prefix may be used.

Most symbolics used in the ICMMMICT job can be cross-referenced to a corresponding symbolic in ICMMMIC1 and other jobs, and the actual value to be used. Table 7-8 provides a cross-reference with the values we used.

*Table 7-8 Previously used symbolics cross-reference*

ICMMMICx symbolic	ICMMMICT Symbolic	Our value
Attributes		
?2000?	?02000?	01022
?2001?	?02001?	01023
?2002?	?02002?	01024
?2003?	?02003?	01025
?2004?	?02004?	01026
?2005?	?02005?	01027
?2006?	?02006?	01028
?2007?	?02007?	01029
?2008?	?02008?	01030
?2009?	?02009?	01031
?2010?	?02010?	01032
?2011?	?02011?	01033
?2012?	?02012?	01034
?2013?	?02013?	01035
?2014?	?02014?	01036

ICMMMICx symbolic	ICMMMICT Symbolic	Our value
?2015?	?02015?	01037
?2016?	?02016?	01038
?2017?	?02017?	01039
?2018?	?02018?	01040
?2019?	?02019?	01041
?2020?	?02020?	01042
?2021?	?02021?	01043
?2022?	?02022?	01044
?2023?	?02023?	01045
?2024?	?02024?	01046
Item type IDs		
?IT2000?	?02000001?	1004
?IT2001?	?02001001?	1023
Item type view IDs		
?ITV2000?	?02000001?	1004
?ITV2001?	?02001001?	1023
Component type IDs		
?CT2000?	?02000001?	1007
?CT2001?	?02001001?	1008
?CT2002?	?02002001?	1009
?CT2003?	?02003001?	1010
Component type view IDs		
?CTV2000?	?02000001?	1007
?CTV2001?	?02001001?	1008
?CTV2002?	?02002001?	1009
?CTV2003?	?02003001?	1010
?CTV2004?	?02004001?	1011

ICMMMICx symbolic	ICMMMICT Symbolic	Our value
Component view IDs		
?CV2000?	?02000001?	1007
?CV2001?	?02001001?	1008
?CV2002?	?02002001?	1009
?CV2003?	?02003001?	1010
?CV2004?	?02004001?	1011
Library Server DB2 table name		
?IMCUT02002001?	?02002001?	ICMUT01009001
?IMCUT02003001?	?02003001?	ICMUT01010001

The ICMMMIC4 loads the resource tables in the Content Manager Library Server database. There are additional symbolics that are not associated with, nor used in, other jobs. Table 7-9 has our values for unique symbolics.

Table 7-9 Unique symbolics in ICMMMIC4

Symbolic	Our value	Description
?ICMUT00300001?	ICMUT00300001?	Base resource table
?ICMUT00304001?	ICMUT00304001?	Annotation resource table

## 7.2.3 Installing Content Manager on z/OS

In our scenario, we install Content Manager for z/OS on the same LPAR where our ImagePlus is running. We have z/OS V1.4 and a single DB2 V7 subsystem.

### Library Server installation

At a high level, the steps we followed to install Library Server are:

1. Prepare SMP/E environment.
2. Copy members ICMALLOC, ICMDDDEF, ICMREC, ICMAPP, and ICMACC from the F6 SMP/E relfile.

Note, in our scenario, our migration was performed prior to general availability and the actual relfile may have changed. See the Library Server Program Directory for the correct information when you perform your installation.

3. Customize and run ICMALLOC.

This creates the Library Server target and distribution data sets.

4. Customize and run ICMDDDEF.  
This defines the data sets for the applicable data definitions for the target and distribution zones.
5. Customize and run ICMREC.  
This receives the Library Server SMP/E data into the SMP/E global zone.
6. Customize and run ICMAPP.  
This applies the Library Server SMP/E data into the SMP/E target zone.
7. Customize and run ICMMLSCR in the SICMINS1 data set.  
This creates the Library Server DB2 storage group and database.
8. Customize and run ICMMLSLD in the SICMINS1 data set.  
This loads the Library Server DB2 database with initial data.
9. Customize ICMMBIND in the SICMINS1 data set.  
This performs the Library Server DB2 binds.
10. Customize ICMMLSBD in the SICMINS1 data set.  
This binds the Library Server DB2 packages and plans.  
  
Note, ensure that the ICMMBIND DD statement points to the ICMMBIND member and data set customized in step 9.
11. Customize and run ICMMLSGT in the SICMINS1 data set.  
This grants appropriate authority to the plans bound in step 10.
12. Customize and run ICMMACL in the SICMINS1 data set.  
This compiles the system-defined access control lists.
13. From a Windows workstation where the Content Manager V8.3 system administration client is installed, go to a command prompt window. From the %CMBICMROOT% sub-directory, issue:  
  
**cmcfgls -t comtypes**  
  
Note: You are prompted for applicable information for accessing your CM/390Library Server. When completed, all applicable information has been stored for the generation of dynamic SQL by the DB2 stored procedures for system-defined entities.
14. From the same sub-directory as in step 13, issue:  
  
**cmcfgls -t predefs**  
  
Note: You are prompted for applicable information for accessing your CM/390Library Server. When completed, creation of the system-defined item types, such as NOINDEX, is complete.

This completes the Library Server installation.

## Resource Manager installation

The V8.3 Resource Manager can be installed in the same SMP/E zones as the Library Server.

At a high level, the steps we followed to install Resource Manager were:

1. Copy members ICMRALLO, ICMRDDEF, ICMRREC, ICMRAPP, ICMRACC, and ICMISMKD from the F2 SMP/E relfile.

Note, in our scenario, our migration was performed prior to general availability and the actual relfile may have changed. See the Resource Manager Program Directory for the correct information when you perform your installation.

2. Customize and run ICMRALLO.

This creates the Resource Manager target and distribution data sets.

3. Customize and run ICMISMKD.

This creates the HFS directory for the Resource Manager CGI program, which is the executable module for the HTTP Server running for the Resource Manager.

4. Customize and run ICMRDDEF.

This defines the data sets for the applicable data definitions for the target and distribution zones.

5. Customize and run ICMRREC.

This receives the Resource Manager SMP/E data into the SMP/E global zone.

6. Customize and run ICMRAPP.

7. This applies the Resource Manager SMP/E data into the SMP/E target zone.

8. Customize and run ICMMRMCR in the SICMINS2 data set.

This creates the Resource Manager DB2 database.

9. Customize and run ICMMRMLD in the SICMINS2 data set.

This loads the Resource Manager DB2 database with initial data.

10. Customize and run ICMMRMBD in the SICMINS2 data set.

This performs the Resource Manager DB2 binds.

11. Customize and run ICMMRMGT in the SICMINS2 data set.

This grants appropriate authority to the plans bound in step 10.

## Accepting installation

After installing Library Server and Resource Manager, validate the installation by importing and retrieving documents. If you can import documents, search for them, and display them on the window without any errors, the installation is considered successful.

Run ICMACC for the Library Server and ICMRACC for the Resource Manager to perform the SMP/E accept.

## 7.2.4 Performing migration

Once Content Manager for z/OS V8.3 has been installed and all post-installation actions have been completed, we have a fully functioning Content Manager environment. Note, at this point, we have not defined any attributes or item types beyond what the installation process does.

We are now ready to perform migration.

### Migration summary

The following is an overview of the steps involved in migration:

1. Copy members to a new partitioned data set.  
The members include all ICMMMIX, ICMMMIFx, and ICMMMIPx.
2. Unload data from DB2 tables.  
This involves customizing and running the jobs ICMMMIF1 through ICMMMIF9, and ICMMMIFA.
3. Process the data.  
This involves customizing and running the ICMMMIP1 through ICMMMIP4 jobs.
4. Define tables and attributes in the new environment, and load data.  
This involves customizing and running the jobs ICMMMICT, ICMMMIC0 through ICMMMIC9, and ICMMMICA jobs (with the exception of ICMMMIC5, which does not exist).

### Step-by-step migration

To customize and run the migration jobs, you need to follow these steps:

1. Copy all ICMMMIX, ICMMMIFx, and ICMMMIPx members in the SICMMIN1 data set to a new partitioned data set to be customized to run the migration.

**Important:** Several of the ICMMPx jobs have SELECTs with “ORDER BY” clauses. *Do not remove these clauses.* Removing them will cause unpredictable results in the output of the ICMMPx jobs.

2. Unload data from DB2 tables by using the following steps:

a. Customize and run ICMMP1. This unloads the following DB2 tables:

- ENTTTAPR - IPFAF/FWA application profile table
- ENTTTUPR - IPFAF/FWA user profile table
- ENTTTFTY - IPFAF/FWA folder type table
- ENTTTFTB - IPFAF/FWA file tab table
- ENTTTFRM - IPFAF/FWA form number table
- EYPTSYMB - IPFAF/API symbolic names table
- EYPTCOLL - IPFAF/API collection name table
- IDOBJCTB - IODM object characteristics table

**Note:** Steps UNLOAD1, UNLOAD2, and UNLOAD5 will complete with COND CODE 0004 and all other steps will complete with COND CODE 0000. The 0004 is normal for the indicated steps because qualified SELECTs, not SELECT \*, are used.

For the IPFAF/FWA tables, we extract APPLIDCD 01. For the IPFAF/API and IODM tables, we extract everything.

b. Before running the remaining ICMMPx jobs, you need to allocate a partitioned data set for the SYSPUNCH DD statement. The RECFM must be FB and the LRECL must be 80.

In our scenario, we allocate ICMV830.ICMPUNCH.

c. Customize and run ICMMP2. This unloads the IPFAF/API folder table.

In our scenario, we specify EYPTFOLD00.

If you perform an *incremental migration*, where you migrate in stages, you need to add additional SELECT criteria to limit the unload to what you want.

For example, if you only want to migrate folders that were created in the year 2003, you need to add:

CRTE DATE >= '2003-01-01' and CRTE DATE <= '2003-12-31'

d. Customize and run ICMMP3. This unloads the IPFAF/API secondary index table.

In our scenario, we specify EYPTSNDX00. Although normally we do not use secondary indexes, we added several secondary indexes to verify that they are properly migrated.

If you perform an *incremental migration*, you need to qualify the SELECT statement in this job to reflect the same criteria as in the ICMMMIF2 job.

- e. Customize and run ICMMMIF4. This unloads the IPFAF/API folder note table.

In our scenario, we specify EYPTNOTE00. Although we do not use folder notes, we added several to verify that they are properly migrated.

If you perform an *incremental migration*, you need to qualify the SELECT statement in this job to reflect the same criteria as in the ICMMMIF2 job.

- f. Customize and run ICMMMIF5. This unloads the IPFAF/API object table.

In our scenario, we specify EYPTOBJT00.

If you perform an *incremental migration*, you need to qualify the SELECT statement in this job to reflect criteria that coincides with the criteria in the ICMMMIF2 job.

**Note:** The SELECT statement in ICMMMIF5 includes the USERSTAT=0 clause.

A USERSTAT of 0 means the document is active in IPFAF. Any other USERSTAT value is considered logically deleted. This means that the object exists in OAM, but is not returned in a list folder contents request.

IPFAF/FWA sets USERSTAT to 1 when a user deletes a document.

If you want to include objects that have been logically deleted, the USERSTAT clause should be removed.

If this is an incremental migration, we recommend you set the USERSTAT to 2 when the migration is complete, so that SELECTs in subsequent runnings of this job will not include objects already migrated. In addition, if you need to switch back to ImagePlus for any reason, you simply change all USERSTAT=2 to 0 and you're back in business.

- g. Customize and run ICMMMIF6. This unloads the IPFAF/API versions table.

In our scenario, we specify EYPTVERS00. Although we do not use object versioning, we added several versions to verify that they are properly migrated.



If you perform an *incremental migration*, you need to qualify the SELECT statement in this job to reflect the same criteria as in the ICMIMIF5 job.

- h. Optionally, customize and run the ICMIMIF7 job. This unloads the IPFAF/API object name table, EYPTONAMxx. Use of this table is very uncommon.

In our scenario, we allow IPFAF to build the object name. Since we do not use the object name table, we elect not to run this job.

- i. Customize and run ICMIMIF8. This unloads the OAM directory table for the storage group in which the annotations are stored.

In our scenario, we specify GROUP53. Although we do not use annotations, we added several annotations to verify that they are properly migrated.

Note: Content Manager requires that annotations be indexed in the Library Server, but ImagePlus does not have the same requirement in IPFAF.

When ImagePlus is used to annotate an object, the base object name is used with a different low level qualifier. By reviewing the SQL, you can see that only A%X objects, where % is a wildcard, are unloaded. The migration utilities pair the annotations with their base objects for indexing into Content Manager.

- j. Customize and run ICMIMIF9. This unloads the IPFAF/API events table. In our scenario, we specify EYPTVNT00.

To be processed properly by the load job, the fields selected need to remain as specified; but it is not necessary to migrate all events. The select criteria can be modified to specify the applicable criteria for the event records that you want to migrate.

- k. Customize and run ICMIMIFA. This unloads the object size information from the OAM directory tables. You can add whatever selection criteria is appropriate for the data you are migrating.

**Note:** If you are only migrating a subset of the documents in your repository, there is no problem with unloading the object size for all documents, because the OAM entries for documents that are not migrated will just be ignored. If, however, your repository is extremely large, and you want to conserve DASD, in this interim step, you can limit what is extracted.

For example, if you are only migrating documents that were stored over a specific period of time, you could qualify the SELECT to only unload the information for that set of objects.

3. Once all of the unload jobs are completed, it is time to process that data as follows:

a. Customize and run ICMMP1. This processes the following data:

- The IPFAF/API Collection table, EYPTCOLL
- The IPFAF/FWA User Profile table, ENTTTUPR
- The IPFAF/API Symbolic Names table, EYPTSYMB

ICMMP1 has three steps. Each step has control data members that need to be customized before running this job. The control data member source is in the SICMMIN1 data set. The three steps are:

- i. Step ICMMP1 with its control data member, ICMCOLL
- ii. Step ICMMP2 with its control data member, ICMUSER
- iii. Step ICMMP3 with its control data member, ICMRM

Table 7-10 defines the contents of control data member, ICMCOLL.

Table 7-10 ICMMP1: Step ICMMP1 with control data member ICMCOLL

Starting column	Ending column	Default value	Parameter description
1	4	0001	This parameter associates the collection names to be processed with the Resource Manager that accesses that collection. If you use only one Resource Manager which references the same OAM that ImagePlus does, leaving the default is fine. This is what we do in our scenario. If you have a separate Resource Manager for your new Content Manager environment and a new Resource Manager for your migrated ImagePlus data, you need to determine which Resource Manager code to use. This can be done by querying the ICMMPRESOURCEMGR table.
6	9	0002	Prefetch Indicator

Table 7-11 defines the contents of control data member, ICMUSER.

Table 7-11 ICMMP1: Step ICMMP2 control data member ICMUSER

Parameter	Default	Parameter description
ActivateFlag	1	The following values are valid: 0 - If user is inactive, bypass the user. 1 - If user is inactive, activate the user and define to CM.
UserPrivSet	0001	All users processed in this run will be assigned this set of privileges.

Parameter	Default	Parameter description
GrantPrivSet	0001	All users processed in this run will have authority to grant these privileges.
DfltACLCode	0003	All users processed in this run will be assigned this access control list.
DfltRMCode	0001	All users processed in this run will be assigned this Resource Manager as their default.
DfltCollCode	0001	All users processed in this run will be assigned this collection as their default for storing objects.
DomainID	0001	All users processed in this run will be assigned to this domain. The implementation is with a single domain. If you want to implement multiple domains, you need to consider how you want to distribute your users among domains.

Table 7-12 defines the contents of control data member, ICMRM.

Table 7-12 ICMMP1: Step ICMMP3 control data member ICMRM

Starting column	Ending column	Default value	Parameter description
1	4	TSI1	The IODM ID from the EYPTSYMB unload.
6	9	0001	The next available Resource Manager code. Note, each Resource Manager is defined with a unique code in the Library Server database. In our scenario, we use only one Resource Manager which uses the same OAM as our IODM does. If you want multiple Resource Managers, you need to query the ICMSTRESOURCEMGR to determine the next available code.
11	80	1.2.3.4	The TCP/IP address of the Resource Manager. A host name could be used as an alternative.

**Note:** We are only migrating one IODM. If your environment has multiple IODMs, you can specify multiple rows in this member. If you use a single Resource Manager, and there is no need to define additional entries, this step should not be executed.

- b. Customize and run ICMMP2. This job processes the ImagePlus folders and their related parts, secondary indexes, and folder notes. The job has a single step with a control data member, ICMFOLD.

ICMFOLD is in the SICMMIN1 data set and needs to be customized before running the ICMMP2 job.

Table 7-13 defines the contents of the control data member, ICMFOLD.

Table 7-13 ICMMP2: Control data member ICMFOLD

Parameter	Our value	Parameter description
SegmentID	001	Segment identifier for the environment being migrated to. In our scenario, we use the default value.
LibraryID	001	Library identifier for the environment being migrated to. In our scenario, we use the default value.
PartitionID	001	Partition identifier for the environment being migrated to. In our scenario, we use the default value.
LibServID	001	Library Server identifier for the environment being migrated to. In our scenario, we use the default value.
ACLCode	0003	Access Control List for the folders processed in this run. We use the same ACL for all our folders. If you need to use multiple ACLs, you can either make multiple runs or change what you want in the output data set or in DB2 after storing the data.
RestartFlag	0	This parameter has the following possible values: 0: Indicates an initial run. 1: Is used for a restart. If this job is previously aborted rather than starting from the beginning, you can restart at the point of failure by specifying a 1. You also need to specify the Restart FLDRTKN in this case.
Restart FLDRTKN		This field should be blank if RestartFlag is 0. If RestartFlag is 1, the folder token of the first folder to be processed must be provided.
Folder CompTypeID	01007	Folder component type identifier. See Table 7-5 on page 192 for the value we use in our scenario.
Note CompTypeID	01008	Folder note component type identifier. See Table 7-5 on page 192 for the value we use in our scenario.
ItemTypeID	01004	Folder item type identifier. See Table 7-7 on page 194 for the value we use in our scenario.

**Note:** Table 7-10, Table 7-11, and Table 7-12 contain the default value column. The default value information may be used as is.

Table 7-13 and Table 7-14 contain the symbolics that need to be replaced. We include the “Our value” column to give you an example of the values you can use for your scenario.

In Table 7-13, for the first four entries, SegmentID, LibraryID, PartitionID, and LibServID, the defaults should be used. Modifying these values would involve a fairly complex migration scenario and discussion of these values is beyond the scope of this IBM Redbook. An IBM IT Specialist should be contacted to discuss the implications of changing the default values.

- c. Customize and run ICMMP3. This job processes the ImagePlus objects and related parts such as file tabs, form numbers, and annotations. The job has a single step which has a control data member, ICMOBJT.

The control data member ICMOBJT is in the SICMMIN1 data set and needs to be customized before running the ICMMP3 job.

Table 7-14 defines the contents of control data member, ICMOBJT.

Table 7-14 ICMMP3: Control data member ICMOBJT

Parameter	Our value	Parameter description
RMCode	1	Resource Manager code for the Resource Manager that manages the objects being migrated. In our scenario, because we migrate only one IODM, which is defined as our only Resource Manager, we take the default. Note: If you migrate more than one IODM that requires multiple Resource Managers being defined, you need to do a phased migration, since you can only refer to a single Resource Manager in each run.
PreFetchInd	0	This parameter is not currently used. In our scenario, we just use the default value.
LibServerID	1	Multiple Library Servers can use the same database. To separate them, each Library Server is assigned a unique identifier in the ICMSTSYSCONTOL table, the LIBRARYSERVERID attribute. In our scenario, we only have one and use the default value.

Parameter	Our value	Parameter description
PartitionID	1	If DB2 partitioning is used, the partition that the data in this run would go to is specified here. In our scenario, we only have one partition. If you spread the data over multiple partitions, you need to do a <b>phased</b> migration. In this case, you need to determine how to split the data between partitions.
SegmentID	1	This value should always be 1.
LibraryID	1	This value should always be 1.
ACLCode	3	Access Control List for the documents/objects being processed in this run. Note: If you want to assign different ACLs to some documents, you can either change it after migration using SQL or do a phased migration, since you can only refer to a single ACL in each run.
UserID	IPMIG	User ID logged as the creator. Note, the default value is ICMADMIN.
SmsCollCode	0	Not currently used. The collection code in the IPFAF/API object table, EYPTOBJTxx, is used.
RestartFlag	0	Not currently used.
RestartTime	9999-99-99-99.99.99.999999	Not currently used.
ItemTypeID	2001	Document item type ID. See Table 7-7 on page 194 for the value we use in our scenario.
CompTypeID	2002	Document component type ID. See Table 7-5 on page 192 for the value we use in our scenario.
ChdCompTypeID	2003	Document child component type ID. See Table 7-5 on page 192 for the value we use in our scenario.
ResItemTypeID	300	Resource item type ID. Always 300.
ResCompTypeID	300	Resource component type ID. Always 300.
AnnItemTypeID	304	Annotation item type ID. Always 304.

Parameter	Our value	Parameter description
AnnCompTypeID	304	Annotation component type ID. Always 304.
PrintFlag	1	The values for this parameter are common with ICMMP4. See Table 7-15 for descriptions of valid values.
FolderID	IP390MIGRATI ZON	The value stored in the IPFOLDER field is for migrated documents.

**Note:** If you cannot use the default values, you should contact an IBM IT Specialist to discuss potential alternatives.

- d. Customize and run ICMMP4. This is the final job in the processing group before we start loading the data. This job creates the Content Manager link table entries which link the documents to the folders where they are located.

This job has a single control parameter, PrintFlag, that needs to be set. The default is 0.

Table 7-15 defines all the valid values and provides their meanings. This parameter is also used for the ICMMP3 job.

Table 7-15 Valid print parameters for ICMMP3 and ICMMP4

Parameter	Description
0	Only write high-level job details to SYSPRINT.
1	Write all migration details in a readable format to SYSPRINT.
2	Write raw migration data, unformatted to SYSPRINT.

We provide the following sample output with various settings:

- PrintFlag set to 1 in ICMMP3: See Example 7-1.
- PrintFlag set to 2 in ICMMP3: See Example 7-2 on page 211.
- PrintFlag set to 1 in ICMMP4: See Example 7-3 on page 212.
- PrintFlag set to 2 in ICMMP4: See Example 7-4 on page 213.

Example 7-1 Sample output of ICMMP3 SYSPRINT with PrintFlag set to 1

```
Control File 'DD:CNTOBJT' successfully opened
Control File 'DD:CNTOBJT' size = 32768
RMCode      = 1
PreFetchInd = 0
LibServerID = 1
```

```

PartitionID      = 1
LibraryID        = 1
ACLCode          = 3
SmsCollCode      = 0
UserID           = IPMIG
RestartFlag      = 0
RestartTime      = 9999-99-99-99.99.99.999999
ItemTypeID       = 1005
CompTypeID        = 1009
ChdCompTypeID    = 1010
ResItemTypeID    = 300
AnnItemTypeID    = 304
AnnCompTypeID    = 304
PrintFlag        = 1
FolderID         = IP390MIGRATION
File Tab File 'DD:ENTTTFTB' successfully opened
File Tabs 'DD:ENTTTFTB' size = 147
File Tab information '1'
  APPLIDCD        = 1
  LANGID          = 'ENU'
  TABCD           = 1
  FILETAB         = 'AUT   '
  MODUSER         = 'DIRECT '
  TIMECHGD        = '1991-04-29-14.08.37.460910'
File Tab information '2'
  APPLIDCD        = 1
  LANGID          = 'ENU'
  TABCD           = 2
  FILETAB         = 'AUT   '
  MODUSER         = 'DIRECT '
  TIMECHGD        = '1991-04-29-14.08.37.460910'
Object File      'DD:EYPTOBJT' successfully opened
Version File     'DD:EYPTVERS' successfully opened
ObjName File     'DD:EYPTONAM' successfully opened
OAM Dir File     'DD:IDOAMDIR' successfully opened
DocRoot File     'DD:ICMDROOT' successfully opened
Resource File    'DD:ICMRESRC' successfully opened
Annotatn File    'DD:ICMANNOT' successfully opened
DocChild File    'DD:ICMCHILD' successfully opened
Reference File    'DD:ICMREFNC' successfully opened
Items File       'DD:ICMITEMS' successfully opened
ItemVer File     'DD:ICMVERSN' successfully opened
IntermObj File   'DD:ICMINOBJ' successfully opened
Next Document #1 created at '2002-12-19-17.57.06.348425' Version '1'
  CLUSTID         = 19766044
  APPLIDCD        = 1
  FOLDTKN         = '2002-04-19-14.17.44.406679'
  RECVDATE        = '2002-12-19'
  OBJTIME         = '2002-12-19-17.57.06.348425'

```



```

CRTESITE      = 'IODM'
OBJVERS       = 1
ORIGKEPT      = '0'
NAMEFLAG      = '1'
FORMCD        = 2
TABCD         = 2
NUMPAGES      = 1
SECURCL       = '01'
COLLCD        = 50
CODEPAGE      = 37
OBJCLASS      = -32768
STORSITE      = 'IODM'
FILEDATE      = '2002-12-19'
USERDATE      = ''
OBJSTAT       = '0'
USERSTAT      = '0'
TIMECHGD      = '2002-12-19-17.57.36.420047'
FORMNUM       = 'A055'
OBJDESC       = 'TEST NUMBER 9'
USERDATA      = ''

```

#### Document Root Information

```

CompClusterID = 11000000
ComponentID   = 'A04K18A81110A00001'
ItemID        = 'A1001001A04K18A81110A00001'
VersionID     = 1
ACLCode       = 3
SemanticType  = 1
CompKey       = 'A04K18A81110A0000100001'
CreateTS      = '2002-12-19-17.57.06.348425'
CreateUserID  = 'ICMADMIN'
LastChgdTS    = '2002-12-19-17.57.36.420047'
LastChgdUserID = 'ICMADMIN'
APPLIDCD      = 1
RECVDATE      = '2002-12-19'
FILEDATE      = '2002-12-19'
USERDATE      = ''

```

---

#### Example 7-2 Sample output of ICMMP3 SYSPRINT with PrintFlag set to 2

---

```

PrintFlag      = 2
FolderID       = IP390MIGRATION
File Tab File 'DD:ENTTTFTB' successfully opened
File Tabs 'DD:ENTTTFTB' size = 147
File Tabs '1' = 1,'ENU',1,'AUT      ','DIRECT  ','1991-04-29-14.08.37.460910'
File Tabs '2' = 1,'ENU',2,'HOM      ','DIRECT  ','1991-04-29-14.08.51.109808'
File Tabs '3' = 1,'ENU',3,'UMB      ','DIRECT  ','1991-04-29-14.09.08.553495'
Form Number File 'DD:ENTTTFRM' successfully opened
Form Numbers 'DD:ENTTTFRM' size = 2219
FormNumber'1 = 1,'ENU',1,1,'A046','CERTIFICATE OF LIABILITY','DIRECT  ','1991-0

```

```

FormNumber'2 = 1,'ENU',2,2,'A055','HOMEOWNER INFORMATION','DIRECT ','1991-04-2
FormNumber'3 = 1,'ENU',3,1,'DW48','ACCIDENT INFORMATION','DIRECT ','1991-04-29
FormNumber'4 = 1,'ENU',4,1,'ICPF','CORRESPONDENCE','DIRECT ','1991-04-29-14.16
FormNumber'6 = 1,'ENU',6,3,'ABCDEFGHJIJ','CLAIM RELEASE FORM','DIRECT ','1991-0
FormNumber'7 = 1,'ENU',7,1,'B039','NATURAL DISASTER CLAIM','DIRECT ','1991-04-
Characterist.c File 'DD:IDOBJCTB' successfully opened
Document'1' ObjTime'2002-12-19-17.57.06.348425' Version'1'
Object   = 19766044,1,'2002-04-19-14.17.44.406679','2002-12-19','2002-12-19-17.
8,'IODM','2002-12-19','','0','0','2002-12-19-17.57.36.420047','A055','TEST NUMB
DocRoot  = 11000003,'A04K22A91323A00001','A1001001A04K22A91323A00001',1,3,1,'A0
ICMADMIN','2002-12-19-17.57.36.420047','ICMADMIN',1,'2002-12-19','2002-12-19','
N'
Resource = 12000003,'A04K22A91323A00002','A1001001A04K22A91323A00002',1,3,128,'
','ICMADMIN','2004-11-22-09.13.23.000002','ICMADMIN',2,50,1091567616,0,'',193009
DocChild = 13000003,'A04K22A91323A00003','A1001001A04K22A91323A00001',1,'A04K22
A0000000001',0,'',1,'A1001001A04K22A91323A00002',300,'A04K22A91323A00002',300,1
Refernce = 'A1001001A04K2291323A00001',1005,'A04K22A91323A00003',1010,1,'A100,1
,'A04K22A91323A000030010rw01',0
DocItem  = 'A1001001A04K2291323A00001',1,'A04K22A91323A00001',1,3,1,1005,1009,'
ResPart  = 'A1001001A04K2291323A00002',1,'A04K22A91323A00002',1,3,128,300,300,'
End of File error on reading next OAM Directory
Interim  = 1,'A1001001A04K2291323A00001','2002-04-19-14.17.44.406679'
Next Version starts at '2004-08-16-17.07.45.954948'

```

---

*Example 7-3 Sample output of ICMMP4 SYSPRINT with PrintFlag set to 1*

---

```

Control File 'DD:CNTRLINK' successfully opened
Control File 'DD:CNTRLINK' size = 32768
PrintFlag      = 1
IntermObj File [DD:ICMINOBJ] successfully opened
IntermFld File [DD:ICMINFLD] successfully opened
Links File [DD:ICMLINKS] successfully opened
Interim Folder Information, Folder# 1
  FolderToken   = '2002-04-19-14.17.44.406679'
Interim Object Information Doc#: 1
  FolderToken   = '2002-04-19-14.17.44.406679'
Links Information Link#: 1
  TargetItemID  = 'A1001001A04K18A81110A00001'
  SourceItemID  = 'A1001001A04K18A75536A00001'
Interim Object Information Doc#: 2
  FolderToken   = '2002-04-19-14.17.44.406679'
Links Information Link#: 2
  TargetItemID  = 'A1001001A04K18A81111A01438'
  SourceItemID  = 'A1001001A04K18A75536A00001'
Interim Object Information Doc#: 3
  FolderToken   = '2002-04-19-14.17.44.406679'
Links Information Link#: 3
  TargetItemID  = 'A1001001A04K18A81111A01441'
  SourceItemID  = 'A1001001A04K18A75536A00001'

```

```

Interim Object Information Doc#: 4
  FolderToken    = '2002-04-19-14-17.44.406679'
Links Information Link#: 4
  TargetItemID   = 'A1001001A04K18A81111A01444'
  SourceItemID   = 'A1001001A04K18A75536A00001'
Interim Object Information Doc#: 5
  FolderToken    = '2002-04-19-14.17.44.406679'
Links Information Link#: 5
  TargetItemID   = 'A1001001A04K18A81111A01447'
  SourceItemID   = 'A1001001A04K18A75536A00001'
Interim Object Information Doc#: 6
  FolderToken    = '2002-04-19-14.17.44.406679'

```

---

*Example 7-4 Sample output of ICMMP4 SYSPRINT with PrintFlag set to 2*

---

```

Control File 'DD:CNTLLINK' successfully opened
Control File 'DD:CNTLLINK' size = 32768
PrintFlag      = 2
InterObj File [DD:ICMINOBJ] successfully opened
InterFld File [DD:ICMINFLD] successfully opened
Links File [DD:ICMLINKS] successfully opened
IntFldr = 1, 'A1001001A04K18A75536A00001', '2002-04-19-14.17.44.406679'
Interm   = 1, 'A1001001A04K22A91323A00001', '2002-04-19-14.17.44.406679'
Links    =
'2004-11-22-09.13.31.000001', 'A1001001A04K22A91323A00001', 'A1001001A0
Interm   = 1, 'A1001001A04K22A91323A01438', '2002-04-19-14.17.44.406679'
Links    =
'2004-11-22-09.13.31.000002', 'A1001001A04K22A91323A01438', 'A1001001A0
Interm   = 1, 'A1001001A04K22A91323A01441', '2002-04-19-14.17.44.406679'
Links    =
'2004-11-22-09.13.31.000003', 'A1001001A04K22A91323A01441', 'A1001001A0
Interm   = 1, 'A1001001A04K22A91323A01444', '2002-04-19-14.17.44.406679'
Links    =
'2004-11-22-09.13.31.000004', 'A1001001A04K22A91323A01444', 'A1001001A0
Interm   = 1, 'A1001001A04K22A91323A01447', '2002-04-19-14.17.44.406679'

```

---

4. All of the processing jobs are now complete. The next steps are defining the tables and attributes in the Content Manager Library Server database created in 7.2.3, “Installing Content Manager on z/OS” on page 197, and loading the data as follows:
  - a. Run job ICMMP4. This job creates the DB2 tables listed in Table 7-16. Table 7-5 on page 192 has the values we used for symbolic substitution in this job.

**Note:** By default, the tables in ICMMMICT are all created in the ICMLFQ32 table space. All of the ICMUTnnnnnnsss tables can be of considerable size. We recommend defining separate table spaces for each table.

Table 7-16 Tables created in ICMMMICT

Table Symbolic	Comments
?CREATOR?.ICMUT?02000001?	Folder component table
?CREATOR?.ICMUT?02001001?	Folder note component table
?CREATOR?.ICMUT?02002001?	Document root component table
?CREATOR?.ICMUT?02003001?	Document child component table
?CREATOR?.ICMUT?02004001?	Document parts component table

- b. Customize and run ICMMMIC0. This loads the Library Server database tables, ICMSTRESOURCEMGR, ICMSTCOLLNAME, and ICMSTUSERS.

Table 7-4 on page 189 has all the symbolic resolution information needed for this job.

- c. Customize and run ICMMMIC1. This defines the attributes, NLS keywords, component and item views.

Table 7-4 on page 189, Table 7-6 on page 193, and Table 7-7 on page 194 have the symbolic resolution information for this job.

- d. The remaining jobs are ICMMMIC2, ICMMMIC3, ICMMMIC4, ICMMMIC6, ICMMMIC7, ICMMMIC8, ICMMMIC9, and ICMMMICA, which need to run in that sequence. They load the ImagePlus metadata into the appropriate Content Manager DB2 tables. See Table 7-4 on page 189 through Table 7-9 on page 197 for the symbolic resolution information.

Note: There is not an ICMMMIC5 job.

## 7.2.5 Performing post migration activities

After completing our migration, there are certain actions that need to be completed to render the environment usable. These actions are:

- ▶ Run ICMMCACL in the SICMINS1 data set. Even though this job was run after the installation, it must be rerun after completing the migration.
- ▶ From a system administration client, execute:

```
cmcfgls -t comtypes
```

Again, even though this was performed after the installation, it must be rerun after completing the migration.

- If additional OAM collections were added from ImagePlus, the Resource Manager bind ICMMRMBD, in the SICMINS2 data set, needs to be modified to include the OAM packages for the applicable OAM storage groups, and then ICMMRMBD needs to be executed.

If rebinding was necessary, the grant ICMMRMGT, in the SICMINS2 data set, needs to be executed.

## 7.3 Customization alternatives

When migrating from an ImagePlus system to a Content Manager V8 system, there are a number of things that can be changed to make the interface more intuitive and efficient, using the provided utilities with minimal custom effort.

Before performing any of these customization alternatives, ensure you have completed the following:

1. Successfully installed a complete Content Manager V8 on z/OS.
2. Successfully executed ICMMMICT, ICMMMICO, and ICMMMIC1.
3. Successfully executed `cmcfgls -t comtypes` from the system administration client. Even though the job was run after the installation, it must be rerun after step 2.

### 7.3.1 Modifying the display name for attributes

One example would be to change the display names for attributes and item types. For instance, the display name that is used for Folder IDs in the migrated item type is IP FOLDER ID. There are two ways to change this value.

One approach is to modify the DDL and SQL prior to running the migration jobs. This can be risky and should be avoided if at all possible.

The other method is to complete the migration and then go to the system administration client and edit the attribute properties. *We highly recommend this approach since it is a fully controlled process.*

Figure 7-2 shows what is initially displayed.

**Attribute Properties - FOLDID**

Name: \* FOLDID

Display name: \* IP FOLDER ID Translate...

**Attribute type**

- ☐ Character
- ☒ Variable character
- ☐ Short integer
- ☐ Long integer
- ☐ Decimal
- ☐ Double
- ☐ Date
- ☐ Time
- ☐ Time stamp
- ☐ BLOB
- ☐ CLOB

**Character type**

- ☐ Alphabetic (1)
- ☐ Numeric (2)
- ☒ Alphanumeric (3)
- ☐ Extended alphanumeric (4)
- ☐ Other (5)

**Character length**

Minimum: 1

Maximum: 40

OK Cancel Apply Help

*Figure 7-2 Default attribute properties in system administration client*

The display name would be changed as shown in Figure 7-3.

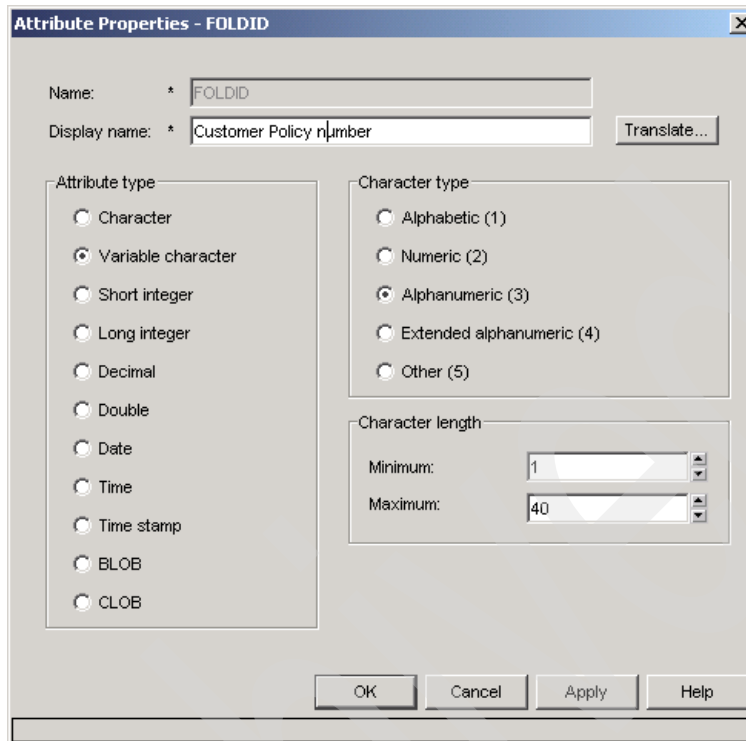


Figure 7-3 Attribute after change in system administration client

If you want to change the name of an item type, the process would be the same with the exception that you would be going to the properties of the item type.

### 7.3.2 Modifying the display name for item types

The name displayed for item types can be changed also. For example, the provided migration utilities give the folder item type a display name of IP390 Folder. You can change it to Customer Policy. You can make this change in the migration DDL and SQL, or change the name after the utilities created the item type. The second approach is simpler, and we highly recommend using this process.

To make the change:

1. Go to the system administration client.
2. Expand the Library Server database. Select **Data Modeling** → **ItemTypes** → **IPFOLDER** (see Figure 7-4).

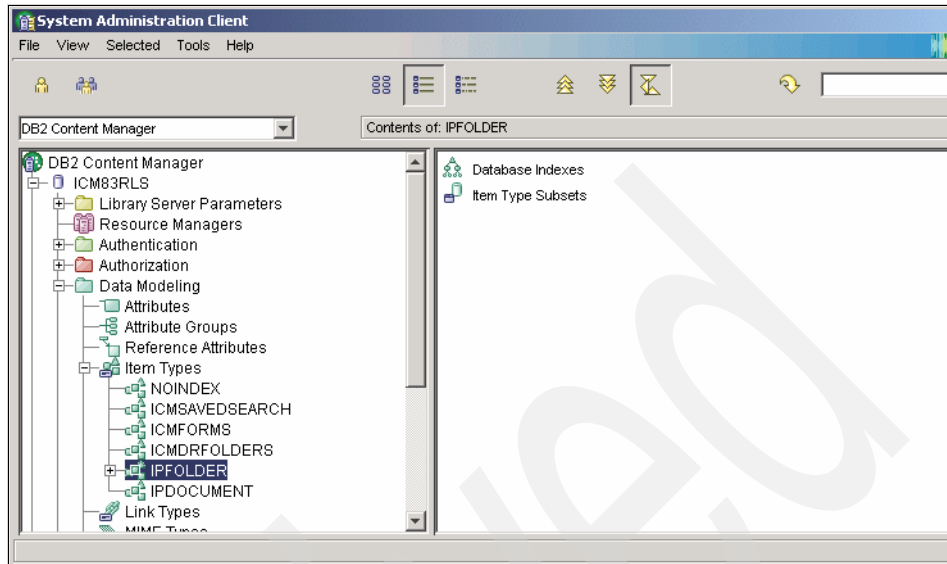


Figure 7-4 IPFOLDER highlighted in system administration client

3. Right-click IPFOLDER and select Properties.

Alternatively, from the pull-down menu, select **Selected** → **Properties** (see Figure 7-5).

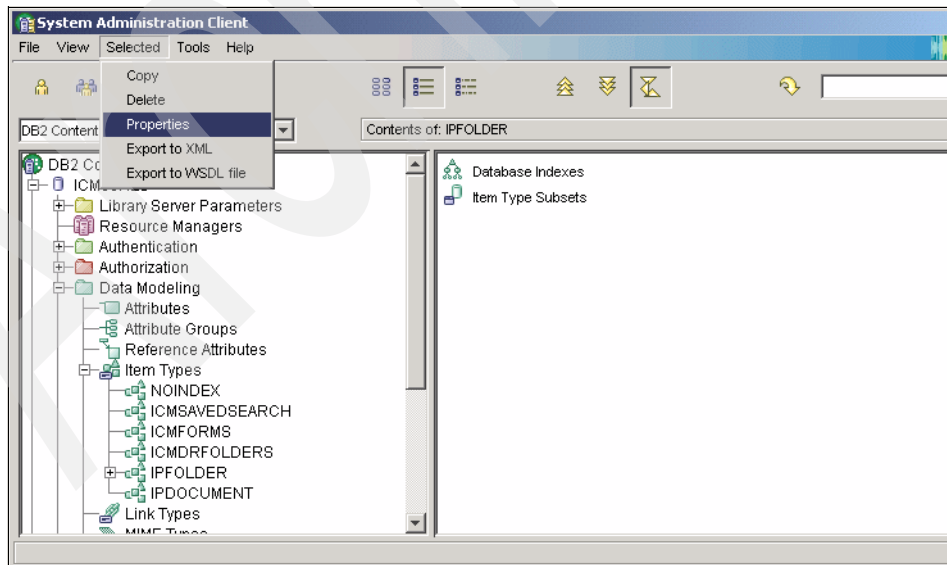


Figure 7-5 Selecting properties for IPFolder item type



4. The item type properties panel appears as shown in Figure 7-6. The default information is provided in the migration utilities.

Change the display name to Customer Policy (see Figure 7-7), and then select **OK** to save the entry.

The dialog box titled "Item Type Properties - IPFOLDER" has a tabbed interface. The "Definition" tab is selected. It contains two text input fields: "Name:" with the value "IPFOLDER" and "Display name:" with the value "IP390 Folder". A "Translate..." button is located to the right of the "Display name" field. The background of the dialog box features a large, faint watermark of a stylized letter 'A'.

Figure 7-6 Default IPFOLDER display name

The dialog box titled "Item Type Properties - IPFOLDER" is shown with the same "Definition" tab selected. In this state, the "Display name:" field now contains the text "Customer Policy". The "Name:" field remains "IPFOLDER". The "Translate..." button is still present. The background watermark 'A' is also visible.

Figure 7-7 Customized folder display name

### 7.3.3 Modifying the attributes in an item type

Attributes can be added once an item type is created. Any added attribute fields would be at the end of the item type. If the position of the added attribute is important and it needs to be somewhere else, a view can be created that places the attribute in the display position as needed.

**Note:** The instructions given in this section assume that the changes are being made before the jobs have been customized. If you have already customized your jobs and replaced the symbolics, you need to determine the corresponding custom values in your jobs.

Attributes cannot be removed from an item type once it is created. The process for removing attributes is considerably more complex and requires great care in performing.

If using the provided migration utilities, with no modifications, actions to accomplish this only require changes to the ICMMICx jobs.

For example, you only use one secondary index on the folder, but the migration utilities create three attributes in the folder item type for secondary indexes. To remove secondary indexes 2 and 3, you need to do the following:

1. Go to the job, ICMRICT, that creates the component table for the folder item type, ?CREATOR?.ICMUT?02000001?. Fields ATTR00000?02013? and ATTR00000?02014? represent the secondary indexes you want to remove. You know that it is these two fields from Table 7-6 on page 193, which give the symbolics and custom values for the attribute IDs. Your first step is therefore to remove these fields.

Note: You can remove the definition of these attributes altogether in the Library Server if you want to, but that would require changing the SQL in several other places, which lends itself to potential errors. The best way to accomplish that would be to go through the System Administrator Client and delete them after performing the migration.

2. The Library Server recognizes the structure of item types and components by definitions in other tables; therefore, it is extremely important to modify these entries as well to keep things in synchronization. The ICMRICT1 job defines the various parts required. The ICMSTCompAttrs and ICMSTCompViewAttrs are the two tables in ICMRICT1 that will be affected. Do the following:
  - a. From Table 7-5 on page 192, the component type ID for the folder root component is 2000. This means that you can go to ICMRICT1 job and find the ICMSTCompAttrs inserts for ?CT2000?. The last two inserts for ?CT2000? are the second and third secondary indexes - attributes ?2013? and ?2014?. These two inserts need to be deleted.
  - b. The ICMSTCompViewAttrs table inserts also need to be modified. Again, the component type ID is 2000. The last two inserts for ?CV2000? are the second and third secondary indexes - attributes ?2013? and ?2014?. These two inserts need to be deleted.
3. Go to the ICMRICT2 job, in which STEP1 loads the component table for the folder item type.

The default DDL for loading the DB2 table can be seen in Example 7-5. Note that for each attribute, the *POSITION* in the input data set is specified.

If the attribute is not referenced, the data will not be loaded. Since we eliminated the attributes from the component table definition in ICMRICT, this works nicely. See the modified DDL in Example 7-6.

*Example 7-5 DDL to load the folder item type component table*

---

```
LOAD DATA LOG NO NOCOPYPEND RESUME YES INDDN SYSRECOO
  INTO TABLE ?CREATOR?.ICMUT?02000001?
    (COMPCLUSTERID      POSITION (1  ) INTEGER,
     COMPONENTID        POSITION (5  ) CHAR(18),
     ITEMID              POSITION (23 ) CHAR(26),
```

```

        VERSIONID          POSITION (49 ) SMALLINT,
        ACLCODE            POSITION (51 ) INTEGER,
        SEMANTICTYPE       POSITION (55 ) INTEGER,
        EXPIRATIONDATE     POSITION (59 ) DATE EXTERNAL
        NULLIF( 69 )='?',
        COMPKEY            POSITION (70 ) CHAR(23),
        CREATETS           POSITION (93 ) TIMESTAMP EXTERNAL,
        CREATEUSERID       POSITION (119 ) CHAR(32),
        LASTCHANGEDTS      POSITION (151 ) TIMESTAMP EXTERNAL,
        LASTCHANGEDUSERID  POSITION (177 ) CHAR(32),
        ATTR00000?02001?   POSITION (209 ) SMALLINT,
ATTR00000?02008?         POSITION (211 ) CHAR(8),
        ATTR00000?02006?   POSITION (219 ) CHAR(2),
        ATTR00000?02007?   POSITION (221 ) DATE EXTERNAL,
        ATTR00000?02009?   POSITION (231 ) TIMESTAMP EXTERNAL,
        ATTR00000?02000?   POSITION (257 ) VARCHAR,
        ATTR00000?02010?   POSITION (299 ) VARCHAR
        NULLIF( 554 )='?',
        ATTR00000?02011?   POSITION (555 ) VARCHAR
        NULLIF( 810 )='?',
        ATTR00000?02012?   POSITION (811 ) VARCHAR
        NULLIF( 853 )='?',
        ATTR00000?02013?   POSITION (854 ) VARCHAR
        NULLIF( 896 )='?',
        ATTR00000?02014?   POSITION (897 ) VARCHAR
        NULLIF( 939 )='?')
ENFORCE NO

```

---

*Example 7-6 Modified DDL to load the folder item type component table*

---

```

LOAD DATA LOG NO NOCOPYPEND RESUME YES INDDN SYSRECOO
INTO TABLE ?CREATOR?.ICMUT?02000001?
(COMPCLUSTERID          POSITION (1 ) INTEGER,
COMPONENTID             POSITION (5 ) CHAR(18),
ITEMID                  POSITION (23 ) CHAR(26),
VERSIONID               POSITION (49 ) SMALLINT,
ACLCODE                 POSITION (51 ) INTEGER,
SEMANTICTYPE            POSITION (55 ) INTEGER,
EXPIRATIONDATE          POSITION (59 ) DATE EXTERNAL
NULLIF( 69 )='?',
COMPKEY                 POSITION (70 ) CHAR(23),
CREATETS                POSITION (93 ) TIMESTAMP EXTERNAL,
CREATEUSERID            POSITION (119 ) CHAR(32),
LASTCHANGEDTS           POSITION (151 ) TIMESTAMP EXTERNAL,
LASTCHANGEDUSERID      POSITION (177 ) CHAR(32),
ATTR00000?02001?       POSITION (209 ) SMALLINT,
ATTR00000?02008?       POSITION (211 ) CHAR(8),
ATTR00000?02006?       POSITION (219 ) CHAR(2),
ATTR00000?02007?       POSITION (221 ) DATE EXTERNAL,

```

```
ATTR00000?02009?    POSITION (231 ) TIMESTAMP EXTERNAL,  
ATTR00000?02000?    POSITION (257 ) VARCHAR,  
ATTR00000?02010?    POSITION (299 ) VARCHAR  
    NULLIF( 554 )='?',  
ATTR00000?02011?    POSITION (555 ) VARCHAR  
    NULLIF( 810 )='?',  
ATTR00000?02012?    POSITION (811 ) VARCHAR  
    NULLIF( 853 )='?')
```

ENFORCE NO

---

# Migration from CM V2.3 to CM V8.3

This chapter provides information about the migration from IBM DB2 Content Manager V2.3 for OS/390 (also known as Content Manager V7.1 for OS/390) to Content Manager for z/OS V8.3.

We cover the following topics:

- ▶ Pre-migration steps
- ▶ Preparing migration jobs
- ▶ Validate the environment
- ▶ Post product migration

## 8.1 Introduction

The migration process consists of migrating the Content Manager V7 metadata to Content Manager V8.3. It does not migrate data objects such as images and documents. The data objects are usually stored in OAM or TSM and it is beyond the scope of what we cover in this IBM Redbook.

Note, this chapter contains the steps that we have gone through during our migration exercises. In certain places, we refer to excerpts from other publications.

You must use the information in this chapter along with the following publications to get a complete detailed picture of the entire migration process:

- ▶ *IBM DB2 Content Manager for z/OS: Planning and Installing Your Content Management System*, GC18-7698
- ▶ *IBM DB2 Content Manager for z/OS: Migrating to Content Manager Version 8 for z/OS*, GC18-7699

Migrating Content Manager V7 to V8.3 consists of four steps:

1. Perform pre-migration steps.
2. Prepare the migration jobs.
3. Run the migration jobs.
4. Validate the environment.

**Important:** The Library Server Logon exit is implemented differently. If your current system uses this exit, you need to rewrite it for Content Manager V8.3 on z/OS.

## 8.2 Pre-migration steps

Before beginning your migration process, you need to make sure all the prerequisites are satisfied. You can find this information in the documentation mentioned in the earlier section.

After ensuring that all prerequisites are met, there are several steps you must complete before the migration process. These steps are summarized as follows:

1. Verify existing data.
2. Upgrade Content Manager.
3. Upgrade DB2.
4. Install Content Manager for z/OS V8.3
5. Verify SMP/E post-installation job, ICMMLSLD.
6. Generate system item and component types.

7. Define additional Resource Managers if applicable.
8. Shut down Library Server.

### **Step 1: Verify existing data**

Because the migration utility does not check the Content Manager database for errors and try to correct them, it is important to verify the data to avoid the migration utility terminating abnormally. The utility may terminate abnormally in these cases:

- ▶ Extra rows in tables AVT00004 and AVT00005.
- ▶ The AVT table definition in DB2 does not match the definition in the Content Manager database.

For information on how to verify these issues and how to solve them, refer to *IBM DB2 Content Manager for z/OS: Migrating to Content Manager Version 8 for z/OS*, GC18-7699.

### **Step 2: Upgrade Content Manager**

The PTF UQ50981 must be applied in your Content Manager V2.3. This PTF will upgrade to Content Manager V7.1 and it is a prerequisite for the migration to Content Manager V8.3. The APAR number from UQ50981 is PQ43125.

### **Step 3: Upgrade DB2**

DB2 V7 or later is required for Content Manager V8.3 and for the migration. If this prerequisite is not satisfied, you need to migrate DB2 to a supported level. To upgrade DB2, refer to *IBM DB2 Installation Guide*, GC26-9936.

### **Step 4: Install Content Manager for z/OS V8.3**

Install Content Manager as described in Chapter 6, “Installation” on page 153, in conjunction with *IBM DB2 Content Manager for z/OS: Planning and Installing Your Content Management System*, GC18-7698.

### **Step 5: Verify SMP/E post-installation job ICMMLSLD**

The post-installation job ICMMLSLD is responsible for loading the default Library Server definitions into the DB2 tables. Refer to Chapter 3, section “Before you begin - Step 4: Verify SMP/E post-installation job ICMMLSLD” on page 26, in *IBM DB2 Content Manager for z/OS: Migrating to Content Manager Version 8 for z/OS*, GC18-7699.

**Note:** In the migration publication mentioned above, Step 4 refers to verifying SMP/E post-installation job ICMMLSLD and Step 5 refers to installing Content Manager for z/OS V8.3.

We switched the order of these two steps. During our migration exercises, we discovered that we must first install Content Manager before we were able to verify the ICMMLSLD job.

### Step 6: Generate system item and component types

Run the following commands from a command prompt in a workstation where the system administration client is installed:

```
<Installation drive \icmroot>\bin\cmbenv81  
<Installation drive \icmroot>\config\cmcfgls -t comtypes -l logfile  
<Installation drive \icmroot>\config\cmcfgls -t predefs -l logfile
```

For more information, go to 6.3, “System administration client installation” on page 169.

### Step 7: Define additional Resource Managers if applicable

If your Content Manager V7 has more than one Object Server, you must define each additional Resource Manager in the new Content Manager V8.3 system using the system administration client.

Do not make any other definitions in Content Manager V8.3.

**Important:** Ensure that there is no user ID with same name in Content Manager V7 and V8.3.

### Step 8: Shut down Library Server

In order to guarantee the database integrity, shut down the CICS region of the Content Manager. Ensure there are no connections to the Content Manager DB2 database.

## 8.3 Preparing migration jobs

You need to customize the migration jobs to conform to your environment. The jobs are in the SICMMIN1 data set. We recommend copying the jobs from the data set SICMMIN1 to a user data set, and executing these jobs from the user data set.



To prepare the migration jobs, there are several steps involved summarized as follows:

1. Map Object Servers with Resource Managers.
2. Customize migration jobs.
3. Create migration tables.
4. Bind the migration package.
5. Migrate system definitions and create user tables.
6. Create directory for export to workstation Object Server.

## Step 1: Map Object Servers with Resource Managers

You need to know which Object Server will be associated with which Resource Manager.

To discover the Object Server's and Resource Manager's names, run the following SQL statements:

```
select objservcode, objservname from ?frn?.frnobjectserver;  
select rmcode, rmname from ?icm?.icmstresourcemgr;
```

The first select is for Object Server, and the second one is for the Resource Manager information.

After you execute the select statements, you must make the pair Object Server - Resource Manager, and add it in the migration job ICMMI70. Use the following SQL statement to insert the pair:

```
insert into ?cm7creator?.cm2icmrms values (?cm7osid?,?cm8rmid?);
```

?CM7OSID? stands for the Content Manager V7 OBJSERVCODE. ?CM8RMID? stands for the Content Manager V8.3 RMCODE.

Follow the instructions in *IBM DB2 Content Manager for z/OS: Migrating to Content Manager Version 8 for z/OS*, GC18-7699.

## Step 2: Customize migration jobs

There are 12 jobs you must customize and two more jobs (ICMMMI7C and ICMMMI8C) if you use multi-valued attributes:

- ▶ ICMMMI70
- ▶ ICMMMI71
- ▶ ICMMMI72
- ▶ ICMMMI73
- ▶ ICMMMI74
- ▶ ICMMMI75
- ▶ ICMMMI76
- ▶ ICMMMI77

- ▶ ICMMMI7A
- ▶ ICMMMI7B
- ▶ ICMMMI7C
- ▶ ICMMMI8B
- ▶ ICMMMI8C

The jobs, ICMMMI72 to ICMMMI77, execute the procedure DSNUPROC to run the DB2 utilities. Depending your environment, the procedure may have another name. You must change it to conform to your environment.

To get a better understanding about these jobs, we recommend reading 9.10, “Run Content Manager V8.3 migration jobs” on page 243, in conjunction with this section.

### **Step 3: Create migration tables**

Run job ICMMMI70. This creates DB2 objects and inserts the IDs of the Object Server-Resource Manager pairs.

### **Step 4: Bind the migration package**

Run job ICMMMI71. This binds the DB2 packages for the migration utility and grants the authorization to execute the plan.

There are 40 bind package statements and one bind plan. You must ensure that all the bind statements complete successfully.

### **Step 5: Migrate system definitions and create user tables**

Run job ICMMMI72. This is the most critical job in the migration step because if this job does not end successfully, you need to recreate the Content Manager database and restart the migration procedure from the beginning.

This job runs the migration utility, ICMPMIDR, and also runs DB2 utilities load and runstats.

If you set in your Library Server WLM procedure //ICMUSQL pointing to ICMMLSQ2, you need to submit the generated jobs manually.

If the job did not run successfully, you will get a message in this format:

xxxxx, yyyy, some text

where:

- ▶ xxxxx is a migration return code.
- ▶ yyyy is the Library Server message (ICMyyyy).

In Appendix D, “Migration error codes and messages” on page 379, we list the error codes and messages for z/OS migration.

## **Step 6: Create directory for export to workstation Object**

### **Server**

Even if you do not have any multiplatform Object Server, the migration utility requires a previously created z/OS UNIX System Services (USS) directory. This USS directory is used by job ICMMMI73.

## **8.4 Running the migration jobs**

Running the migration jobs involves the following steps:

1. Migrate parts data.
2. Migrate WIPITEMS data.
3. Migrate checkout data.
4. Migrate links data.
5. Migrate items data.
6. Verify user tables definitions.
7. Migrate user tables.
8. Export data to workstation object servers.

### **Step 1: Migrate parts data**

Run job ICMMMI73. This job runs the Content Manager migration utility program ICMPMIDR and runs the DB2 load utility.

Check the migration log file, <userID>.CM728MIG.LOG, to ensure the job runs successfully.

Note, the migration utility ICMPMIDR requires the USS directory that was previously created.

### **Step 2: Migrate WIPITEMS data**

Run job ICMMMI74. This job runs the Content Manager migration utility program ICMPMIDR and runs the DB2 load utility.

Check the migration log file, <userID>.CM728MIG.LOG, to ensure the job runs successfully.

### **Step 3: Migrate checkout data**

Run job ICMMMI75. This job runs a select statement and stores the result in a data set that is used as the input for the DB2 load utility.

#### **Step 4: Migrate links data**

Run job ICMMMI76. This job runs the DSNTIAUL DB2 sample program and the load utility.

#### **Step 5: Migrate items data**

Run job ICMMMI77. This job executes DSNTIAUL and the load utility.

#### **Step 6: Verify user tables definition**

Confirm whether or not the user tables are created successfully. You should not proceed to the next step until all user tables are created.

#### **Step 7: Migrate user tables**

Run job ICMMMI7A. This job migrates the user tables. It automatically runs the following jobs:

- ▶ ICMMMI7B
- ▶ ICMMMI8B
- ▶ ICMMMI7C
- ▶ ICMMMI8C

#### **Step 8: Export data to workstation Object Servers**

This task is only required if you have Object Servers defined in multiplatforms.

## **8.5 Validate the environment**

After the migration, you need to check to see if everything was migrated successfully. You can use the Content Manager Client to look for an existing document, import a new document, and retrieve a document.

## **8.6 Post product migration**

As mentioned earlier in this chapter, the Library Server Logon exit is implemented differently. If your current system uses this exit, you need to rewrite it for Content Manager V8.3 on z/OS.

## Migration from multiplatform to z/OS

This chapter describes the considerations and steps involved in migrating a Content Manager V7 system on multiplatforms (such as Windows or AIX) to a Content Manager V8.3 system on z/OS.

We concentrate on the specific issues related to moving a system from multiplatform to z/OS. For a more general description on how to migrate a Content Manager system on z/OS, refer to chapter Chapter 8, “Migration from CM V2.3 to CM V8.3” on page 223.

We cover the following topics:

- ▶ Introduction
- ▶ Things to consider before migration
- ▶ Migration process
- ▶ Database structure
- ▶ Backup
- ▶ Individual migration steps include:
  - Updating existing system
  - Installing new V8.3 system
  - Copying Library Server database to the new system
  - Running migration jobs
  - Migrating Object Server and its objects

## 9.1 Introduction

There can be a number of reasons why you need to move from a Content Manager (CM) system on multiplatforms (such as Windows or AIX) to a z/OS platform. Some of the reasons are:

- ▶ Need to consolidate your systems on the z/OS platform.
- ▶ Merger with another company. Need to migrate the “inherited” multiplatforms system to the corporate standard platform, z/OS.
- ▶ Growth of your Content Manager system makes it relevant to move to the z/OS platform.
- ▶ Requirement to integrate your Line Of Business applications with Content Manager using the new Content Manager z/OS APIs.

Depending on your current environment, you may have to go through a number of steps to bring your existing environment up to date before you can perform a migration. For example, if you still use a VisualInfo™ Version 2.n system, you must upgrade the existing system to a Version 6 or Version 7 level, since there is no migration path to Version 8.n available from the older VisualInfo Versions.

## 9.2 Things to consider before migration

Before embarking on a migration project, you must evaluate the impact on how the system is used today. Things to consider include:

- ▶ Are there any custom applications using the Content Manager APIs that need modification?

Refer to the following IBM Redbooks for detailed discussions on migrating Content Manager API applications:

- *Content Manager Implementation and Migration Cookbook*, SG24-7051
- *Content Manager V8.1 Migration Guide for Multiplatforms*, SG24-6877

- ▶ Are you using server side exits depending on local server data?
- ▶ Are you, for some reason, accessing the Content Manager tables directly on your server (for example, for statistical purposes)?
- ▶ Are you using text search facilities that are not available on z/OS?

These issues are among many others that may have an important impact on your migration plans.

## 9.3 Migration process

The challenge involved in migrating a Content Manager system from multiplatforms to z/OS is that there is no standard out-of-the-box utilities included with the product enabling you to perform such tasks. You have to perform a number of manual steps to accomplish this type of migration.

**Important:** You must have a thorough understanding of the Content Manager system structure before you proceed.

We assume that the starting point is a Content Manager V7 system with the Library Server and Object Server on a multiplatforms platform.

We outline the steps you need to perform in order to migrate your Content Manager system from V7 on multiplatforms (MP) to V8.3 on z/OS:

1. Understand the data structure.  
See 9.4, “Database structure” on page 234.
2. Back up your current Content Manager system on multiplatforms.  
See 9.5, “Backup, backup, backup...” on page 235.
3. Apply maintenance to the current Content Manager V7 system so that the starting point is Library Server V7 and Object Server V7 on multiplatforms.  
See 9.6, “Update existing CM system on multiplatforms” on page 235.
4. Review and document various database names on both the multiplatforms and z/OS systems.  
See 9.7, “Review database naming conventions” on page 235.
5. Install your target Content Manager V8.3 system on z/OS.  
See 9.8, “Install Content Manager V8.3 system on z/OS” on page 236.
6. Create Library Server V7 database on z/OS and copy data from multiplatforms to z/OS.  
See 9.9, “Copy CM V7.1 Library Server database to z/OS” on page 236.
7. Use the migration tools shipped with Content Manager V8.3 on z/OS to migrate the Content Manager V7 database to new Content Manager V8.3 format.  
See 9.10, “Run Content Manager V8.3 migration jobs” on page 243.
8. Migrate the Object Server on multiplatforms to a Resource Manager V8.3 on multiplatforms.  
See 9.11, “Migrate Object Server to Resource Manager” on page 254.

9. Migrate the new Content Manager V8.3 system to use the V8.3 Resource Manager on z/OS and migrate the objects from the V8.3 Resource Manager on multiplatforms to the one on z/OS.

See 9.12, “Resource Manager and objects migration” on page 260.

In the remaining sections of this chapter, we discuss each of the steps of the migration process. Some of the activities are already documented in detail in other chapters or other IBM Redbooks. We do not copy all of this existing information into this chapter, but rather point you to the existing information and concentrate on the issues that are specific to the cross-platform migration.

## 9.4 Database structure

When planning the cross-platform migration, it is important to first look at the differences in implementation of the Content Manager components on the two types of platforms and how data is stored.

### 9.4.1 Library Server database

The structures of the Library Server database for Content Manager V7 on Multiplatforms and Content Manager V2.3 on z/OS are similar apart from the naming of the tables.

On Multiplatforms, the table names for the system tables used by the Library Server all start with SBT (for example, SBTPATRONS). For Content Manager V2.3 on z/OS, these tables have names starting with FRN (for example, FRNPATRONS).

Because of the table naming differences, we must modify the SQLs used in all the migration jobs referring to the Content Manager Library Server tables as described for each migration job in the following sections.

In addition to the Library Server system tables, the Library Server has all the tables holding the item type data. These tables have names such as AVTnnnnn and MVTnnnnn. The naming of these tables is the same on multiplatforms as on z/OS. The table names are built based on the internal number assigned to the index class and multi-value attributes.

### 9.4.2 Resource Manager database and object storage

Due to the different ways objects are stored on the Windows platform (using files and folders in the LBOSDATA directory) and on z/OS (using OAM where data is



stored in DB2 tables), the implementation of the Resource Manager database is very different between the two types of the systems.

Because of these differences, there is no option to directly export the objects from multiplatforms and import them to z/OS. We therefore use the ability of the Content Manager V8.3 Resource Manager to migrate the objects from one Resource Manager to another based on the definition of the management class.

## 9.5 Backup, backup, backup...

We cannot stress the importance of this enough: Make sure you have a full backup of your Content Manager system on multiplatforms before you start the migration process.

We recommend a full system backup, consisting of the following components:

- ▶ Library Server database
- ▶ Object Server databases
- ▶ LBOSDATA folders
- ▶ STAGING areas

These backups will be your only way back if something goes wrong during your migration efforts. You must also verify that your backup is successful before proceeding with migration.

## 9.6 Update existing CM system on multiplatforms

As a starting point before doing the migration, you must bring your existing Content Manager system on multiplatforms to a level supported by the migration utilities shipped with Content Manager V8.3. This involves updating your Content Manager system on multiplatforms to V7.

In order to migrate the Content Manager V7 Object Server to the Content Manager version V8.3 Resource Manager, you also need to meet any prerequisites outlined for Content Manager V8.3 on multiplatforms such as using DB2 V8 with the stated fixpack levels.

## 9.7 Review database naming conventions

Since you will be working with a number of databases and systems, this is a good time to review and document the names of all the databases (for V7 and V8.3) that are involved in multiplatforms and z/OS environment.

For our scenario, we list the names we used in Table 9-1. These names are used in all the examples we show in this chapter.

*Table 9-1 List of the names used in the examples in this chapter*

Name	Describes
DK07321	Schema name of Content Manager V7 Library Server tables
ITSO7LIB	Library Server V7 database copied to z/OS
ITSO8LS	Content Manager V8.3 Library Server on z/OS
ITSO8RM	Content Manager V8.3 Resource Manager on z/OS
RMPALM	Content Manager V8.3 Resource Manager on multiplatforms

## 9.8 Install Content Manager V8.3 system on z/OS

The migration programs that migrate your Content Manager V7 to V8.3 depends on a fully installed and operating Content Manager V8.3 system on z/OS. This includes components such as the Library Server database, the necessary programs, and the Workload Manager (WLM) procedures.

Before you begin the migration process, you need to install a Content Manager V8.3 system (both Library Server and Resource Manager) on the target z/OS system.

Do not define the users, new item types, attributes, or any other Library Server resources to the new Content Manager V8.3 system yet. You may run the risk of creating duplicate user IDs or other entries which may prevent proper migration.

To verify the successful installation of the system, you can log on to the system using the system administration client and the Windows client. During installation, you also defined the NOINDEX item type. To verify access to your Resource Manager on z/OS, we recommend that you import a document to the NOINDEX item type, perform a search on the NOINDEX item type, retrieve, and display the document. This is a quick way to verify that you can communicate with your Resource Manager HTTP server and that the access to OAM works.

## 9.9 Copy CM V7.1 Library Server database to z/OS

To migrate a Content Manager V7 system on multiplatforms to a Content Manager V8.3 system on z/OS, one of the steps involved is to copy the Content Manager V7 Library Server database from the multiplatforms to the target z/OS

system. Once that is done, you can run the migration utility to migrate the V7 Library Server database to V8.3 on the same z/OS platform.

In this section, we show you how to copy Content Manager V7 Library Server database to z/OS. The steps are summarized as follows:

1. Use DB2LOOK to extract DB2 table definitions.
2. Create Content Manager V2.3 database on z/OS.
3. Catalog CM V7 z/OS database on workstation.
4. Create Content Manager V2.3 tables on z/OS.
5. Unload data from multiplatforms Library Server database.
6. Load data into z/OS database.

We also discuss alternative ways of moving a DB2 database.

### 9.9.1 Use DB2LOOK to extract DB2 table definitions

DB2LOOK is a tool that can extract the necessary SQLs to define the objects defined in a DB2 database. The definitions can be saved as SQL statements in flat ASCII text files and then be used to define the database objects elsewhere.

Use the following command, on the V7 Library Server database on multiplatforms, to create a file with the SQLs necessary to recreate the V7 tables and views in the new database on z/OS:

```
DB2LOOK -d database_name -z schema_name -e -o database.DDL
```

Review the output from DB2LOOK written to the output file database.DDL using a preferred text editor. You need to modify the content before using it to create the Content Manager V7 tables on z/OS:

- ▶ Change the CONNECT statement at the beginning of the file to connect to your new z/OS database.
- ▶ Update DB2 table space names. DB2 on multiplatforms may use names longer than eight characters. You must change these table space names to be a maximum of eight characters. In addition, the table space names must be prefixed with the name of the database in which they are created.

In our scenario, we made the following changes:

```
from: IN USERSPACE "USERSPACE1"  
to: IN USERSPACE "ITS08LIB"."ITSOTS1"
```

- ▶ Update the SET CURRENT SCHEMA = "*schema\_name*" to SET CURRENT SQLID = '*schema\_name*'. Note the use of single quotes:

```
from: SET CURRENT SCHEMA = "DK07321 "  
to: SET CURRENT SQLID = 'DK07321';
```

- ▶ Remove all the “ALTER TABLE *table\_name* ADD CONSTRAINT ...” at the end of the generated output file. These definitions are only needed to ensure integrity of the data when running a Library Server system on the tables. The constraints would prevent us from loading data in a later step.

## 9.9.2 Create Content Manager V2.3 database on z/OS

In our scenario, we created the new Content Manager V2.3 database on z/OS using the JCL shown in Example 9-1.

**Note:** The Content Manager product version is called V2.3. There is actually no official Content Manager V7 on z/OS. The V7 in the database name is used to create the conceptual naming link to the V7 database copied from multiplatforms in contrast to the V8 of the new Content Manager system.

We use V2.3 and V7 interchangeably in this redbook.

Depending on your V7 system setup, substitute the appropriate values for the following values we used in our scenario for the JCL:

- ▶ USRT009 - Replace with your DB2 user authorized to create the database.
- ▶ ITS07LIB - Replace with your V7 Library Server database name.
- ▶ ITS07STO - Replace with your V7 storage group.
- ▶ ITS07TS1 - Replace with your V7 table space.
- ▶ CMST53 - Replace with your volume name.
- ▶ ICMITSO - Replace with your catalog name.

*Example 9-1 JCL used to create CM V2.3 Library Server database on z/OS*

```
//ICMRDB JOB (XXXX), 'CREADB ', NOTIFY=&SYSUID,
//          MSGLEVEL=(1,1), MSGCLASS=0
//JOB LIB   DD DISP=SHR, DSN=SYS1.DB2L.SDSNLOAD
//*
//* CREATE Content Manager 2.3 DATABASE AND TABLE SPACES
//*
//ICMTBSP EXEC PGM=IKJEFT01, DYNAMNBR=20, REGION=1024K
//SYSTSPRT DD SYSOUT=*
//SYS PRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSTSIN DD *
DSN SYSTEM(DB1B)
RUN PROGRAM(DSNTIAD) PLAN(DSNTIAD) -
      LIB('DSN.DB1B.RUNLIB.LOAD')
//SYSIN DD *
SET CURRENT SQLID = 'USRT009';

DROP DATABASE ITS07LIB;
```

```

COMMIT;

DROP STOGROUP ITS07ST0;
COMMIT;

CREATE STOGROUP ITS07ST0 VOLUMES ('CMST53') VCAT ICMITS0;
COMMIT;

CREATE DATABASE ITS07LIB STOGROUP ITS07ST0;
COMMIT;

CREATE TABLESPACE ITS07TS1
IN ITS07LIB
USING STOGROUP ITS07ST0
PRIQTY 5000
SECQTY 1000
FREEPAGE 1
PCTFREE 5
BUFFERPOOL BP32K
LOCKSIZE ROW LOCKMAX 0
CLOSE NO;
//

```

---

### 9.9.3 Catalog CM V7 z/OS database on workstation

After creating the database on z/OS, we need to catalog the database on the PC workstation using the commands shown in Example 9-2.

Depending on your system setup, substitute the appropriate values for the following parameters and values we specified in the example:

- ▶ `node_id` - This is a symbolic name. Choose any value you like.
- ▶ `9.30.128.228` - Replace this with the IP address of your z/OS system.
- ▶ `8060` - Replace this with the port name that your DB2 on z/OS uses.
- ▶ `db0b` - This is the name of our DB2 location on z/OS. If in doubt, look in the DB2 MSTR task on z/OS to see the location value.
- ▶ `ITS07LIB` - Replace this with your V7 Library Server database name.

*Example 9-2 Catalog Library Server V7 z/OS database on PC*

```

DB2 CATALOG TCP/IP NODE node_id REMOTE 9.30.128.228 SERVER 8060
DB2 CATALOG DATABASE db0b AS ITS07LIB AT NODE node_id AUTHENTICATION SERVER

```

---

This makes the Content Manager V7 Library Server database accessible to us under the name ITS07LIB.

## 9.9.4 Create Content Manager V2.3 tables on z/OS

Once the V7 Library Server database is cataloged, you can now create the DB2 tables and indexes in the new database on z/OS.

If you run the table creation from your workstation, you must first define the node and database alias to your z/OS database as illustrated in the previous section. To create the tables and indexes based on the output file `database.ddl` created with DB2LOOK in 9.9.1, “Use DB2LOOK to extract DB2 table definitions” on page 237, use the following command:

```
DB2 -tvf database.ddl > output.txt
```

Review the output file for errors.

You may see the following error in the output file if you did not make the suggested modifications to the `database.ddl` file:

```
DB21034E The command was processed as an SQL statement because it was not a
valid Command Line Processor command. During SQL processing it returned:
SQL0969N There is no message text corresponding to SQL error "-625" in the
message file on this workstation. The error was returned from module
"DSNXIAB1" with original tokens "DK07321.SBTVIEWDEFS". SQLSTATE=55014
```

This is acceptable. It is caused by statements such as this ALTER statement if they are still in the `database.ddl` file:

```
ALTER TABLE "DK07321 "."SBTNLSLANGUAGES" ADD PRIMARY KEY ("LANGUAGECODE");
```

## 9.9.5 Unload data from multiplatforms Library Server database

You are now ready to extract data from the Content Manager V7 Library Server database from multiplatforms.

Since there may be other tables in the database that are not related to Content Manager V7, extract only the tables that are relevant based on the table names.

We extract tables with the following generic names:

- ▶ SBT\*
- ▶ AVT\*
- ▶ MVT\*

Use the DB2MOVE command with the EXPORT option to extract data from the tables. Since the DB2MOVE command only accepts one generic name per call, you need to extract data in three steps into separate folders.

Example 9-3 shows the commands we used in our scenario. ITSOLIB points to the V7 Library Server database on the multiplatforms. Replace it with the name

you use in your environment and make sure you can connect to it before unloading data from it.

*Example 9-3 Export data from the CM V7 Library Server tables*

---

```
CD ..\SBTdata
DB2MOVE ITSOLIB EXPORT -tn SBT*
CD ..\AVTdata
DB2MOVE ITSOLIB EXPORT -tn AVT*
CD ..\MVTdata
db2move ITSOLIB EXPORT -tn MVT*
```

---

After running the export commands, each folder will now contain a DB2MOVE.LST file listing the names of the exported tables and to which xxx.IXF file the data is exported.

Some sample content of the first few lines from the DB2MOVE.LST file from the \SBT folder is shown in Example 9-4. In addition, the folder will contain all the xxx.IXF files with the unloaded data.

*Example 9-4 Sample content of a DB2MOVE.LST file*

---

```
!"DK07321 ". "SBTNLSLANGUAGES"!tab1.ixf!tab1.msg!
!"DK07321 ". "SBTNLSKEYWORDS"!tab2.ixf!tab2.msg!
!"DK07321 ". "SBTCNTL"!tab3.ixf!tab3.msg!
!"DK07321 ". "SBTOBJECTSERVER"!tab4.ixf!tab4.msg!
!"DK07321 ". "SBTCOLLNAME"!tab5.ixf!tab5.msg!
!"DK07321 ". "SBTACCESSCODES"!tab6.ixf!tab6.msg!
!"DK07321 ". "SBTPRIVILEGES"!tab7.ixf!tab7.msg!
!"DK07321 ". "SBTATTRDEFS"!tab8.ixf!tab8.msg!
```

---

## 9.9.6 Load data into z/OS database

Once you unload the data from Content Manager V7 Library Server in multiplatforms, we are ready to load this data into the z/OS database.

To load data, use DB2MOVE command with IMPORT option.

Example 9-5 shows the commands we use to import the data to the z/OS Library Server database in our scenario. We run the DB2MOVE command from each of the folders to which we unloaded the data. The database name ITS07LIB is the name we used when cataloging the database. See 9.9.3, “Catalog CM V7 z/OS database on workstation” on page 239.

*Example 9-5 Import data to Content Manager V2.3 tables on z/OS*

---

```
CD ..\SBTdata
DB2MOVE ITS07LIB IMPORT -io INSERT -u userid -p password
```

```
CD ..\AVTdata
DB2MOVE ITS07LIB IMPORT -io INSERT -u userid -p password
CD ..\MVTdata
DB2MOVE ITS07LIB IMPORT -io INSERT -u userid -p password
```

---

The IMPORT command reads the list of tables to which to import from the DB2MOVE.LST file. If you have a need to split the import process into multiple steps, you can edit this file and include only those tables you want to import in each step. Make sure all the tables are uploaded.

### 9.9.7 Status at this point

We have now created a copy of Content Manager V7 Library Server database on z/OS and loaded all the Library Server data from multiplatforms into the Content Manager tables on z/OS platform.

We are now ready to run the migration jobs on z/OS which will migrate various Content Manager V2.3 tables into the respective Content Manager V8.3 tables.

### 9.9.8 Alternative ways of moving a DB2 database

In our scenario, the amount of data in the Library Server database that we have to migrate is fairly limited. We have no problems using the DB2MOVE method.

If, however, your Library Server database is of a significant size, you may wish to look into alternative ways of moving the database from your multiplatform server to your target z/OS system.

One of the methods we recommend you look into as an alternative is DB2 Cross Loader. This DB2 utility is designed to copy large amount of data across the various DB2 platforms and can be used to copy a large Content Manager Library Server database from multiplatforms to z/OS.

Refer to the following books for an in-depth description of the DB2 Cross Loader utility and additional alternatives for moving large amounts of data:

- ▶ *Moving Data Across the DB2 Family*, SG24-6905
- ▶ *DB2 UDB V8 z/OS Utility Guide and Reference*, SC18-7427
- ▶ *DB2 UDB V7 z/OS Utility Guide and Reference*, SC26-9945

**Note:** At the time of our writing this IBM Redbook, there is a performance PTF about to be completed that will significantly improve the cross loader performance in the DB2 V8 environment. If you are going to move a large database, check the availability of PTF PQ90263.



No matter which method you decide to use in order to move your Content Manager Library Server database, the tasks described in the following sections are the same once you successfully copied the V7 Library Server database from multiplatforms to z/OS.

## 9.10 Run Content Manager V8.3 migration jobs

Tailor and run the migration jobs from data set xxx.SICMMIN1 following the steps outlined in manual *IBM DB2 Content Manager for z/OS: Migrating to Content Manager Version 8 for z/OS*, GC18-7699.

We assume that you follow the instructions as laid out in Chapter 3 “Migrating from Content Manager for OS/390 to Content Manager for z/OS” of the manual mentioned above. We also assume and recommend that you refer to Chapter 8, “Migration from CM V2.3 to CM V8.3” on page 223 for additional reference.

In this section, we emphasize the special considerations related to migrating the Library Server database we just copied from multiplatforms to z/OS.

We list the jobs in the sequence described in the migration manual and refer to the job names used in that manual:

- ▶ ICMMMI70
- ▶ ICMMMI71
- ▶ ICMMMI72
- ▶ ICMMMI73
- ▶ ICMMMI74
- ▶ ICMMMI75
- ▶ ICMMMI76
- ▶ ICMMMI77
- ▶ ICMMMI7x and ICMMMI8x

### 9.10.1 ICMMMI70

The ICMMMI70 job creates a set of DB2 tables in your Content Manager V2.3 Library Server database. These tables are used to assist the migration programs in mapping and unloading data from the Content Manager V2.3 Library Server tables to the Content Manager V8.3 tables.

Before running the ICMMMI70 job, you must use the system administration client to define the Content Manager V7 Object Servers that are being migrated to the new Content Manager V8 Resource Managers (RM) and referenced by your Library Server (LS).

We assume that you defined the Content Manager V8.3 Resource Manager you are going to migrate to when you installed the Content Manager V8.3 Library Server on z/OS.

When defining the new Resource Managers to the Content Manager V8.3 Library Server, entries that are created in the DB2 table ICMSTResourceMgr and each Resource Manager are assigned a unique internal reference number that is used to link the stored Library Server items to a given Resource Manager. In order to migrate the Library Server V7 tables to the new V8.3 tables, the migration program must know how to map the old V7 Object Servers to the new V8.3 Resource Managers since the internal reference numbers used by the Library Server may change (the reference numbers are created based on the sequence they are defined to the Content Manager Library Server).

In the last step of job ICMMMI70, you provide the mapping between the V7 Object Servers and the V8.3 Resource Managers using SQL INSERT statements to the ICM2ICMRMS migration table. The migration programs use this table to perform the mapping of the Resource Manager number.

You can use the SQLs shown in Example 9-6 to extract the current definitions from your Library Server tables.

---

*Example 9-6 Extract OS and Resource Manager codes from the CM tables*

---

```
-- List version 7 Object Server definitions:
Select ObjServCode, ObjServName from DK07321.SBTObjectServer;

-- List version 8 Resource Manager definitions:
Select RMCODE, RMName from ITS08ADM.ICMSTResourceMgr;
```

---

Example 9-7 shows the output we get from our Content Manager systems.

---

*Example 9-7 List of Object Servers and Resource Managers*

---

```
OBJSERVCODE OBJSERVNAME
-----
0 OBJSERVO
1 ITS00BJ1

2 record(s) selected.

RMCODE RMNAME
-----
0 reserved
1 ITS08RM
2 ITS00BJ1
```

3 record(s) selected.

---

Based on the list of definitions from Content Manager V7 and Content Manager V8.3 tables, create the input statements for job ICM MMI70.

In our scenario, we use the statements shown in Example 9-8.

*Example 9-8 Update the CM2ICMRMS migration table*

---

```
//SYSIN DD *  
-- Map the default entries in the CM tables:  
INSERT INTO DK07321.CM2ICMRMS VALUES (0, 0);  
-- Map V7 definition for OS 1 to v8 definition RM 2  
INSERT INTO DK07321.CM2ICMRMS VALUES (1, 2);
```

---

Before submitting the ICM MMI70 job, make the necessary adjustments to the JCL as outlined in the prolog of the job.

The SQL in step S2 of the job creates a number of DB2 VIEWS on the Content Manager V7 tables. The job references these tables with names starting with FRN (for example FRNPATRONS). Change all references to DB2 table names starting with FRN to start with SBT since all our Content Manager V7 table names start with SBT.

## 9.10.2 ICM MMI71

Run job ICM MMI71 to bind the migration programs.

Adjust the job as outlined in the prolog of the job. Pay special notice to updating the QUALIFIER parameter in all the BIND PACKAGE statements. The BIND in the job must reference the schema of your Content Manager V7 Library Server tables.

As shown in Example 9-9, you specify the schema name of the V7 Library Server tables. The example shows only one of the BIND statements. There are about 40 of them in the job. Make sure you change them all.

*Example 9-9 The BIND in job ICM MMI71*

---

```
BIND PACKAGE (ITSOM728) -  
  MEMBER(ICMMMI7R) -  
  QUALIFIER(DK07321) -  
  OWNER(USRT009) -  
  VALIDATE(BIND) -  
  ISOLATION(UR) -  
  RELEASE(COMMIT) -
```

### 9.10.3 ICMMMI72

This is the first step in extracting data from your Content Manager V7 Library Server and loading it into Content Manager V8.3 Library Server tables.

The following data is extracted from the V7 Library Server tables and loaded into the V8.3 Library Server tables:

- ▶ User definitions
- ▶ User group definitions
- ▶ Access lists

In addition, the SQLs to create tables for the item types (“index classes” in Content Manager V7 terminology) are also generated when this job is run.

**Important:** When customizing the WLM environment used by the V8.3 Library Server, you have a choice between three ways of generating and submitting jobs to execute the SQLs that define the item type tables.

Before submitting the ICMMMI72 job, we strongly recommend that you update your WLM procedure to use the option described as ICMMLSQ2 in the chapter “Customize and copy the workload manager procedure” of the installation manual. This option saves the generated jobs in a PDS and you have the opportunity to review and modify the definitions as necessary.

Alternatively, there may be many jobs submitted all at once (about two per Content Manager V7 index class and additional jobs for MVT tables). You may have a challenge following up on all of the generated output.

#### Before running the job

You must ensure that you have no users defined in the Content Manager V8.3 Library Server that are going to be migrated from the Content Manager V7 Library Server. If you run the migration to a newly installed Content Manager V8.3 system, you only need to verify that the system administration user and the connect user defined during installation are not also used in your Content Manager V7 system.

If the names of the users defined in Content Manager V8.3 also exist in your V7 system, you must either:

- ▶ Log on to your Content Manager V7 system before migration and remove the duplicate users.
- ▶ Redefine the names of Content Manager V8.3 users.

## Modify the job

**Important:** If duplicate user IDs are encountered during migration, the migration job will fail.

In addition to modifying the job as outlined in the prolog of the JCL, you must change all references of the SQL statements to Content Manager V7 tables with FRNxxx names to point your SBTxxx tables.

## Submit and check status

Submit the job.

After the job completes, review the following:

- ▶ Job log
- ▶ Log file xxx.CM728MIG.LOG
- ▶ Jobs created in the PDS used by the WLM of the Library Server

In case you have problems, see the instruction in the migration manual for steps to take before resubmitting the job.

One side effect, in case this job terminates abnormally in step S2 while executing the DB2 load utility, can be the loaded tables such as the user table ICMSTUsers can be locked or in a check pending state. If at this stage you try to log on to your Content Manager V8.3 Library Server, you will get strange error messages rejecting your logon attempt with the administration user.

In case step S2 terminates abnormally, you need to run ICMMMI80 with token ?UID? changed to ICMMMI72 to terminate the load. Normally this will unlock the lock and you can rerun the load step after correcting the problem.

If the table space is in a check pending state, you can try the REPAIR utility. Example 9-10 shows a sample job that repairs table space ICMSFQ04 for database ICMMLSDb.

*Example 9-10 Sample job to repair table space ICMSFQ04 for database ICMMLSDb*

---

```
//NOCCPEND JOB ,MSGLEVEL=1,MSGCLASS=0,NOTIFY=&SYSUID
//STEP1 EXEC DSNUPROC,UID='ICM',
//      UTPROC='',
//      SYSTEM='DB2C'
//SYSIN DD *
      REPAIR SET TABLESPACE ICMMLSDb.ICMSFQ04 NOCOPYPEND
```

---

Please refer to the chapter “Troubleshooting Job ICMMMI72” in *IBM DB2 Content Manager for z/OS: Migrating to Content Manager Version 8 for z/OS*,

GC18-7699, for current information on how to recover from an error situation in the ICMMMI72 job.

## Running the jobs created by Library Server WLM

**Attention:** At some point before you proceed with migration job ICMMMI7A as described in 9.10.9, “ICMMMI7x and ICMMMI8x” on page 251, you must successfully run *all* the jobs created by the WLM in the designated PDS. These jobs create the item type DB2 tables and they are used as the target tables when migrating the index classes to the new Content Manager V8.3 system.

### 9.10.4 ICMMMI73

This ICMMMI73 job runs the migration utility ICMPMIDR and also runs DB2 utilities load and runstats.

The ICMMMI73 job performs the following tasks:

- ▶ Extract information on parts from Content Manager V7 Library Server tables.
- ▶ Load data into the Content Manager V8.3 Library Server parts tables.
- ▶ Create files needed by the Resource Manager migration on multiplatforms server.
- ▶ Create file used as input when migrating the AVTnnnnn item type tables.

#### Modify the job

In addition to modifying the job as outlined in the prolog of the JCL, you must change all references of the SQL statements to Content Manager V7 tables with FRNxxx names to point your SBTxxx tables.

#### Submit and check status

Submit the job and check the job log for successful completion.

Status information is also written to the xxx.CM728MIG.LOG file. Review this file and check for errors.

#### Output from the job: the DEL files

Upon completion of the job, go to the z/OS UNIX System Services (USS) environment and check the output files created in the USS folder referenced in the SYSIN of step S1.

You should see one or two files for each V7 Object Server on multiplatforms that is referenced by the Library Server.

The files have the names:

- ▶ *object\_server\_name*OBJS.DEL
- ▶ *object\_server\_name*REPLICAS.DEL

In our environment, we only have one Object Server defined and it is not set up for replication; so we see one file, ITSOOBJ1OBJS.DEL.

The DEL files must be copied to the machines where the Content Manager V7 Object Servers are migrated to Content Manager V8. The files are used as input when creating the Content Manager V8 Resource Manager database (which is described in 9.11, “Migrate Object Server to Resource Manager” on page 254). You must complete the migration steps performed by job ICMMMI73 successfully before you begin to migrate your Object Servers.

Transfer the files using your preferred FTP tool to the workstations where you are performing the Content Manager V7 Object Server to Content Manager V8 Resource Manager migration. If you are operating an environment with multiple distributed Resource Managers, you only need to transfer the specific xxxxxxxxOBJS.DEL and xxxxxxxxREPLICAS.DEL files relevant to a given Resource Manager to the workstation of the Resource Manager.

Also note, an output file xxx.CHILD.SYS is created by the job. This file is used as input when migrating the Content Manager V7 Library Server AVTnnnnn tables running the ICMMMI7A jobs. So the xxx.CHILD.SYS data set must be kept until all the AVTnnnnn tables have been successfully migrated.

### 9.10.5 ICMMMI74

The ICMMMI74 job performs the following tasks:

- ▶ Extract WIPITEMS related data into files.
- ▶ Load extracted data into Content Manager V8 Library Server tables.

#### **Modify the job**

In addition to modifying the job as outlined in the prolog of the JCL, you must change all references of the SQL statements to Content Manager V7 tables with FRNxxx names to point your SBTxxx tables.

#### **Submit and check status**

Submit the job and check the job log for successful completion.

Status information is also written to the xxx.CM728MIG.LOG file. Review this file and check for errors.

### **Output from the job**

Data is loaded into the Content Manager V8 Library Server tables.

## **9.10.6 ICMMMI75**

The ICMMI75 job performs the following tasks:

- ▶ Extract information on checked out documents.
- ▶ Load extracted data into Content Manager V8 Library Server tables.

### **Modify the job**

In addition to modifying the job as outlined in the prolog of the JCL, you must change all references of the SQL statements to Content Manager V8 tables with FRNxxx names to point your SBTxxx tables.

### **Submit and check status**

Submit the job and check the job log for successful completion.

Status information is also written to the xxx.CM728MIG.LOG file. Review this file and check for errors.

### **Output from the job**

Data is loaded into the Content Manager V8 Library Server tables.

## **9.10.7 ICMMMI76**

The ICMMI76 job performs the following tasks:

- ▶ Extract information on linked documents.
- ▶ Load extracted data into Content Manager V8 Library Server tables.

### **Modify the job**

In addition to modifying the job as outlined in the prolog of the JCL, you must change all references of the SQL statements to Content Manager V7 tables with FRNxxx names to point your SBTxxx tables.

### **Submit and check status**

Submit the job and check the job log for successful completion.

No status information is written to the xxx.CM728MIG.LOG file.



## Output from the job

Data is loaded into the Content Manager V8 Library Server tables.

### 9.10.8 ICM MMI77

The ICM MMI77 job performs the following tasks:

- ▶ Extract item data from SBTITEMS table.
- ▶ Data is loaded into Content Manager V8 items table.

## Modify the job

In addition to modifying the job as outlined in the prolog of the JCL, you must change all references of the SQL statements to Content Manager V7 tables with FRNxxx names to point your SBTxxx tables.

## Submit and check status

Submit the job and check the job log for successful completion.

No status information is written to the xxx.CM728MIG.LOG file.

## Output from the job

Data is loaded into the Content Manager V8 Library Server table, ICMSTItems001001.

### 9.10.9 ICM MMI7x and ICM MMI8x

The migration of the Content Manager V7 Library Server item type tables (AVTnnnnn and MVTnnnnn) is done through a series of jobs:

- ▶ ICM MMI7A: Start the migration process for the next AVT or MVT table.
- ▶ ICM MMI7x: Extract data from a Content Manager V7 index class table.
- ▶ ICM MMI8x: Load data into the new Content Manager V8 item type tables.

## Before you run any of these jobs

In this step, you migrate data into your item type tables ICMUTnnnnn001 where *nnnnn* is a sequence number (larger than 1000) that relates to the item type.

You have no control over the sequence in which the item types are migrated. Before starting the migration process, you must have completed the creation of *all* the item type tables defined by the jobs generated in the steps described in 9.10.3, “ICM MMI72” on page 246.

## The sequence of the jobs

The jobs perform the migration tasks in the following way:

1. Submit the first ICMMMI7A job which does the following:
  - a. The ICMMMI7A job reads information from CM2ICMCTL table to find the next AVT or MVT table to migrate.
  - b. It then generates the next migration job by reading the sample JCL from member ICMMMI7B for an AVT table or from member ICMMMI7C for a MVT table.
  - c. The JCL is scanned for special place holders such as the table name to read from and the place holders are replaced with values for the AVT or MVT table to be processed.
  - d. The ICMMMI7B or ICMMMI7C job is now submitted for execution.
  - e. A field in the CM2ICMCTL table is updated for the AVT table we just processed to prevent it from being processed again. If the migration of an AVT or MVT table fails, you have to manually reset this flag to process it again.
2. The ICMMMI7B and ICMMMI7C migration jobs extract data from the Content Manager V7 Library Server table and build input files used to load the Content Manager V8 item type tables.

As a last step, the job reads sample load job from member ICMMMI8B/C, replaces place holders in the job, and submits the job.
3. The ICMMMI8B and ICMMMI8C migration jobs run and load data into the Content Manager V8 item type tables.

Upon successful completion of the load process, the last step of the job submits a new ICMMMI7A job and the migration process continues by determining the next table to migrate.

As you can see from the above sequence, only one job executes at any given time, and as a last step, the next job is built and submitted.

If you would like to control the migration process and not just have it automatically run through all the index classes in one long sequence, we recommend that you comment out the last step in jobs ICMMMI8B and ICMMMI8C that submits the next ICMMMI7A job.

If you do this, the migration process stops after migrating one AVT table. You will have the opportunity to review the output from each migration run. Afterwards, you can resume the process by manually submitting a new ICMMMI7A job.

If you decide after the first few migration runs that it works as you expected, you can then reactivate the last step in jobs ICMMMI8B and ICMMMI8C, and the

migration process will now restart itself after each loop processing the next index class table.

### **Modify the jobs**

Before running the ICMMMI7A job for the first time, you must complete the modification of all five jobs: ICMMMI7A, ICMMMI7B, ICMMMI7C, ICMMI8B, and ICMMI8BC.

In addition to modifying the job as outlined in the prolog of the JCL, you must change all references of the SQL statements to Content Manager V7 tables with FRNxxx names to point your SBTxxx tables.

### **Submit and check status**

Submit the job and check the job log for successful completion.

No status information is written to the xxx.CM728MIG.LOG file.

### **Output from the job**

Data is loaded into the Content Manager V8.3 Library Server tables, ICMUTnnnnn001 for each item type.

You can see the mapping between the Content Manager V7 AVTnnnnn tables and the Content Manager V8.3 ICMUTnnnnn001 tables in the migration table DK07321.

### **Halting the migration process**

If you need to interrupt the migration process and the continuous submission of jobs, you need to prevent the next ICMMMI7A job from being submitted.

We do not recommend that you simply cancel the current migration job since this requires you to perform various recovery tasks as described in the migration guide later. Instead, we suggest that you rename the ICMMMI7A member in your xxx.SICMMIN1 data set. This makes the last step in the next ICMMMI8B or ICMMMI8C job fail when it tries to submit the next migration job. This way, you halt the process after the import of data to the ICMUTnnnnn001 table completes successfully, and there is no need to perform any recovery or cleanup activities.

In order to resume the migration process, just rename the job member back to ICMMMI7A and submit the job. This then resumes the migration process with the next AVT or MVT table based on the status information in the migration table CM2ICMCTL.

## Rebuild the component definitions

Upon successful migration of the AVT tables, you must rebuild the component definitions in the Library Server. For previous users of Content Manager V8.2, this is the equivalent of rebuilding the access modules that contained the static SQL to access the item type tables. In the new release of Content Manager V8.3, these access modules are no longer needed. Instead, we execute the command from a workstation to build and prepare a set of dynamic SQL statements that are saved in a Content Manager table:

```
cmcfgls -t comtypes
```

## Refresh DB2 statistics

After loading the ICMUTnnnnn001 tables with data, you should run a DB2 REORG and RUNSTAT on the Content Manager Library Server tables to refresh the DB2 statistics for the Library Server tables.

After this, run the Content Manager V8.3 Library Server BIND job ICMMLSBD.

### 9.10.10 Status for Content Manager 8.3 Library Server migration

We have now completed the migration of the V7 Library Server tables into the new Content Manager V8.3 Library Server database.

Before we can access any data, we need to complete the migration of the Content Manager V7 Object Server into the new Content Manager V8.3 Resource Manager since we cannot access a V7 Object Server from a V8 Library Server.

## 9.11 Migrate Object Server to Resource Manager

In this section, we discuss how to migrate from a Content Manager V7 Object Server to a Content Manager V8.3 Resource Manager.

The installation of the V8.3 Resource Manager and the prerequisite components on a server on multiplatforms is beyond the scope of this chapter. Refer to the installation manual *IBM DB2 Content Manager Enterprise Edition V8.3: Planning and Installing your Content Manager System*, GC27-1332, for a detailed description of the installation process.

Before proceeding, we assume that you have successfully completed the installation of the Content Manager V8.3 Resource Manager on multiplatforms.

### 9.11.1 Tools to migrate Object Server database

We now migrate the Content Manager V7 Object Server database to a new Content Manager V8.3 Resource Manager database on multiplatforms as described in the manual *IBM DB2 Content Manager Enterprise Edition V8.3: Migrating to DB2 Content Manager Version 8*, SC27-1343.

If you have previously performed a Resource Manager migration on multiplatforms using Content Manager V8.2, be aware that the bat file ICMIMPO.bat file has been split into two separate bat files.

Originally, the ICMIMPO.bat is used to take V7 arguments and perform both the export of data from the Object Server and the import of data to the new Resource Manager in one process.

For V8.3, there are two separate files that replace the original function of the ICMIMPO.bat file:

- ▶ ICMIMPO.bat: Export data from the Content Manager V7 Object Server.
- ▶ ICMIMPR.bat (new): Import data to the new V8.3 Resource Manager.

If you look into the bat files, you see that they actually call the same program but use two different options to control whether it is an export or import process.

### 9.11.2 Export Object Server data

As a first step, you must complete the export of data from the Content Manager V7 Object Server using the following command:

```
ICMIMPO CM70Sname CM70Sadmin CM70SAdminpassword CM7tblspc
```

Where:

- ▶ CM70Sadmin is used as a user ID when connecting to DB2 as well as the schema name for the Object Server tables.
- ▶ CM7tblspc must be the name of an existing table space in the Object Server database in which several migration assist tables are created.

Upon successful completion of the command, you will see a number of xxx.DEL files in the migration folder. These files contain the exported data from the Object Server and are used in the next step to load the Resource Manager tables.

### 9.11.3 Load Resource Manager tables

In order to load the Content Manager V8.3 Resource Manager tables, you need the DEL files created in the previous export step *as well as* the DEL files created by the Library Server migration on z/OS. These files have the following names:

- ▶ `object_server_nameOBS.DEL`
- ▶ `object_server_nameREPLICAS.DEL`

See “Output from the job: the DEL files” in section 9.10.4, “ICMMI73” on page 248 where these files are created and how to download them.

We copy the files to the migration folder on the V8.3 Resource Manager server and issue the following command:

```
ICMIMPR CM8RMname CM8RAdmin CM8RAdminpassword
```

Where:

- ▶ `CM8RAdmin` is used as a user ID when connecting to DB2 as well as the schema name for the Resource Manager tables

One of the changes to the Content Manager V8.3 Resource Manager database on multiplatforms is that it contains additional information on objects that was previously kept in the Library Server database. This additional information is contained in the xxx.DEL files we downloaded from the Library Server migration.

After completing the ICMIMPR command, we have finished loading the Content Manager V8.3 Resource Manager tables with data.

### 9.11.4 Update RMSEVER and RMACCESS tables

Before you can access and use the migrated Resource Manager database, there are a few manual steps you need to complete before you can test and validate the migration process.

We have now reached one of the points where you will notice that the migration tools do not yet fully support an automatic cross-platform migration.

The V8.3 Resource Manager database we migrated from the V7 Object Server still contains information about the name, location, and platform type of the original V7 Library Server to which it was connected as well as the user ID and password of the original administration user. We need to change this definition to reference the new Library Server on z/OS.

To accomplish this, you need to update the RMSEVER and RMACCESS tables in the V8.3 Resource Manager database.

## Update RMSEVER table

Query the RMSEVER table to get information on the entry you need to update. Example 9-11 shows the query and Example 9-12 shows the sample output.

*Example 9-11 Get information from RMSEVER table*

```
--
-- Query the server definitions in Resource Manager
--
set current SQLID = 'RMADMIN';

-- Get content of RMServer table
Select
SVR_ServerID          as ID,
SVR_ServerType        as Type,
substr(SVR_ServerName,1,10) as SVR_ServerName,
substr(SVR_HostName,1,12)  as SVR_HostName,
SVR_Port              as SVR_Port,
substr(SVR_Schema,1,9)    as SVR_Schema,
substr(SVR_Path,1,26)     as SVR_Path,
SVR_SerPlatform        as Platform
from RMServer
;
```

*Example 9-12 Sample content from the RMSEVER table*

ID	TYPE	SVR_SERVERNAME	SVR_HOSTNAME	SVR_PORT	SVR_SCHEMA	SVR_PATH	PLATFORM
0	RM	rmpalm	localhost	8080	rmdmin	/icrm/ICMResourceManager	0
1	LS	ITS08LS	9.30.128.228	0	ITS08ADM		S390

2 record(s) selected.

When you look at the sample output in Example 9-12, you see the two servers defined to the Resource Manager: the Resource Manager itself and the Library Server to which it is associated. It is the entry for the Library Server that we need to update. You can use the “Where SVR\_SERVERID = 1” in the SQL to get to the Library Server entry. Note, in Example 9-12, the resulting column label ID corresponds to SVR\_SERVERID field in the RMSEVER table.

Example 9-13 shows the SQL you can use to update the RMSEVER table with the new Library Server information.

*Example 9-13 Update RMSEVER table with new Library Server information*

```
--
-- After migrating the CM7 OS to new CM8 Resource Manager tables
-- we must update the information about the Library Server to which it is
-- connected:
```

```
-- SVR_SERVERNAME : Library Server name
-- SVR_HOSTNAME   : IP address or name where Library Server runs
-- SVR_SCHEMA     : Schema name of Library Server tables
-- SVR_SERVERID   : the Library Server entry we're updating
-- SVR_SERPLATFORM: type of platform server runs on
--
Set current SQLID = 'RMADMIN';

Update RMserver
Set SVR_SERVERNAME = 'ITS08LS',
SVR_HOSTNAME = '9.30.128.228',
SVR_SCHEMA = 'ITS08ADM',
SVR_SERPLATFORM = 'S390'
Where SVR_SERVERID=1
;
```

---

## Update RMACCESS table

You also need to update the information about the user the V8.3 Resource Manager uses to connect to the V8.3 Library Server. This information is kept in the RMACCESS table in the Resource Manager database.

Example 9-14 shows you how to query information in the RMACCESS table. Example 9-15 shows the sample output of the query.

### *Example 9-14 Get information from RMACCESS table*

```
--
-- Query the access user definitions in Resource Manager
--
set current SQLID = 'RMADMIN';

-- Get content of RMAccess table
Select ACC_UserID, ACC_UserName
From RMAccess
;
```

---

### *Example 9-15 Sample content from the RMACCESS table*

```
ACC_USERID ACC_USERNAME
```

```
-----
0 radmin
1 usrt009
```

```
2 record(s) selected.
```

---

In the sample output shown in Example 9-15, we see the current entries in the RMACCESS table. The value in the ACC\_USERID column matches the value in



the SVR\_SERVERID of the RMSERVER table. From this example, it is the row with “ACC\_USERID = 1” we need to update for the Library Server connection user.

Example 9-16 shows how to update the Library Server connection user information.

*Example 9-16 Update RMACCESS table with information on the LS user*

---

```
--  
Set current SQLID = 'RMADMIN';  
  
-- Update information on the userid we ...  
Update RMAccess  
Set ACC_USERNAME = 'USRT009' ,  
ACC_PASSWORD = 'password_value'  
Where ACC_USERID = 1  
;
```

---

### 9.11.5 Migration checkpoint

As this point, you have completed the following tasks:

- ▶ Migrated the Content Manager V7 Library Server on multiplatforms to the Content Manager V8.3 Library Server on z/OS
- ▶ Migrated the Content Manager V7 Object Server on multiplatforms to the Content Manager V8.3 Resource Manager still on multiplatforms

We can now log on to the Content Manager V8.3 Library Server on z/OS, search for documents in the item types, and retrieve documents from the Content Manager V8.3 Resource Manager on multiplatforms.

If the migration plan only called for a migration of the Library Server to z/OS and you wanted to keep the existing Resource Manager on multiplatforms, your migration would be fully completed at this stage and you can skip the remaining sections of this chapter.

In our scenario, we also want to move the objects from the Resource Manager on multiplatforms to the Resource Manager on z/OS and use the Resource Manager on z/OS for all new objects we store in Content Manager V8.3.

To do so, let us continue with the remaining migration processes as described in the following sections.

## 9.12 Resource Manager and objects migration

In this section, we discuss how you can migrate from Resource Manager on multiplatforms to Resource Manager on z/OS. In addition, we discuss how you can move existing objects from Resource Manager on multiplatforms to Resource Manager on z/OS.

We assume you have followed all the steps described in the previous sections.

In order to accomplish Resource Manager and objects migration, you need to:

1. Update V8.3 Library Server definitions.
2. Validate access to Resource Manager.
3. Define z/OS Resource Manager (RM) to multiplatforms RM.
4. Define migration rules.
5. Setup migrator task.
6. Start migration of objects.

### 9.12.1 Update V8.3 Library Server definitions

The definitions for users as well as item types (the former index classes in Content Manager V7) we migrated to the new Library Server on z/OS still contain the old default references to the Resource Manager on multiplatforms. We now update these definitions to point to our new V8.3 Resource Manager on z/OS.

If we do not do this, we will continue storing new objects in the Resource Manager on multiplatforms.

#### Change user defaults

The migrated user definitions still reference the old defaults from the Library Server database on multiplatforms. In our scenario, it still points to RMPALM, the Resource Manager on multiplatforms. See Figure 9-1.

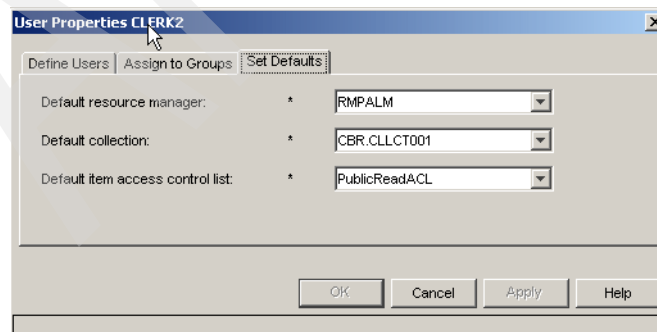


Figure 9-1 User defaults migrated from Library Server on multiplatforms

To ensure that the users from now on use the new Resource Manager on z/OS, update the default settings to point to the new V8.3 Resource Manager on z/OS.

## Change item type definitions

To start storing new objects on V8.3 Resource Manager on z/OS, update the item type definitions to use collections on the z/OS Resource Manager.

Figure 9-2 shows the details, in our scenario, for part type ICMBASE after we selected the line and clicked **Edit**. Figure 9-2 points to V8.3 Resource Manager on Multiplatforms.

Change the *settings* for the part type so that in the future, the part type will be stored on V8.3 Resource Manager on z/OS; otherwise, new objects will continue to be stored on V8.3 Resource Manager on Multiplatforms.

You need to make this change for all the part types in all the item types which point to the Resource Manager on multiplatforms.

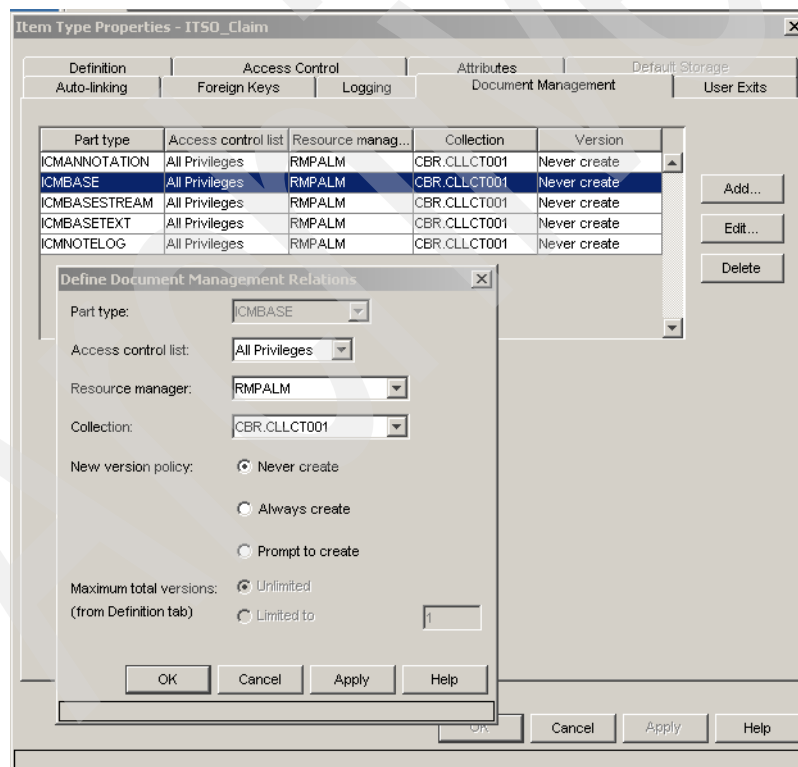


Figure 9-2 Migrated item type definition initially points to RM on multiplatforms

## 9.12.2 Validate access to Resource Manager

Using your Web browser, make sure you can access your Resource Manager on z/OS using the port number and the application name specified in the HTTP configuration file on the host.

In Figure 9-3, we show how we test the connection to our Resource Manager on z/OS by typing the IP address of the host, the port name on which the HTTP server listens, and the case-sensitive string used to call the Resource Manager. The string has the following format:

`http://ip-address:port-number/ICMRMResourceManager`

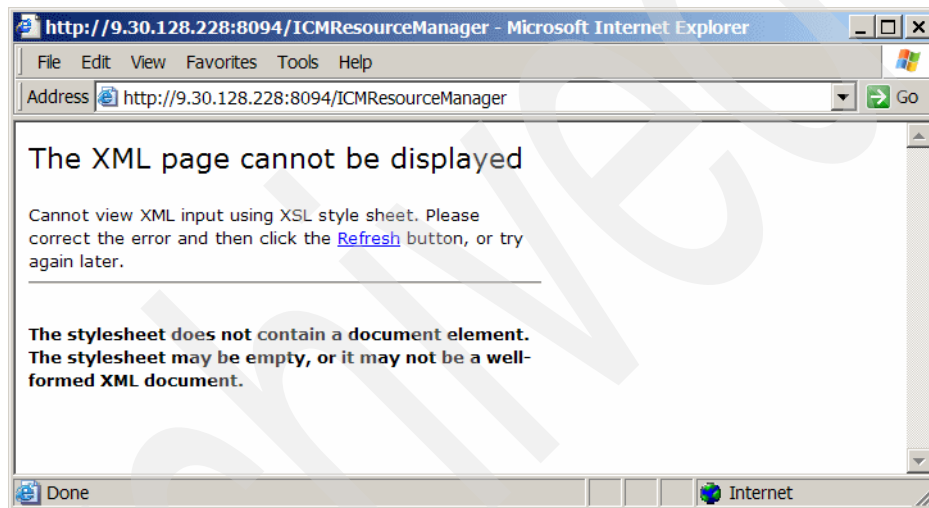


Figure 9-3 Test connection to Resource Manager on z/OS

The message generated by our request is not solid evidence that the Resource Manager is working correctly. Based on the fact that we see an error message saying “The XML page cannot be displayed”, we may interpret this as having received a reply from “something” even though the browser cannot display it.

In Figure 9-4, we show what happens if we type an invalid application string (you can see that the string is almost the same but it is all *lower case*). The result that displays is the same if the port number was incorrect, the HTTP server was inactive, or some other error occurred in the server’s attempt to process your request.

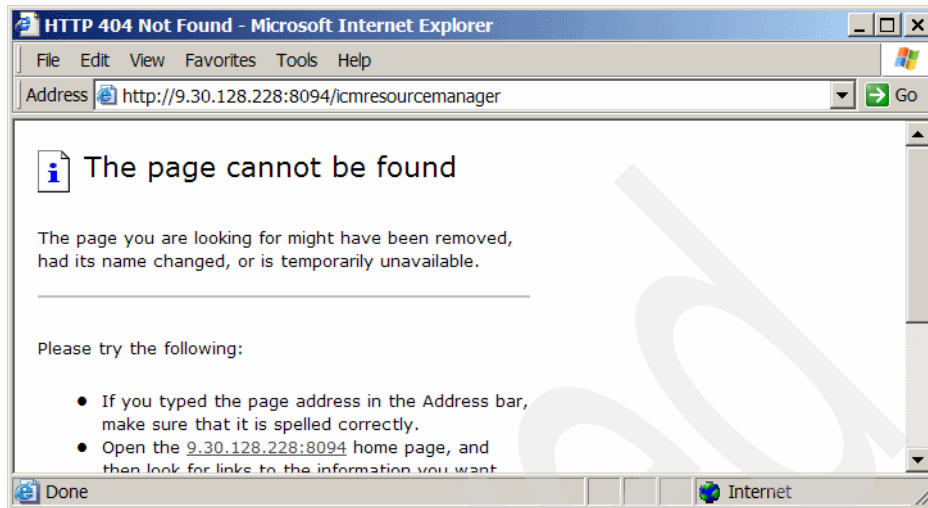


Figure 9-4 Failed request to Resource Manager server on z/OS

There seems to be no other supported way to generate an acknowledgement from the Resource Manager service, but it would be nice if the Resource Manager server program implemented an echo command that could be used to test the connection.

If you try to connect to your Resource Manager on Multiplatforms, you will see a slightly different reply as illustrated in Figure 9-5.

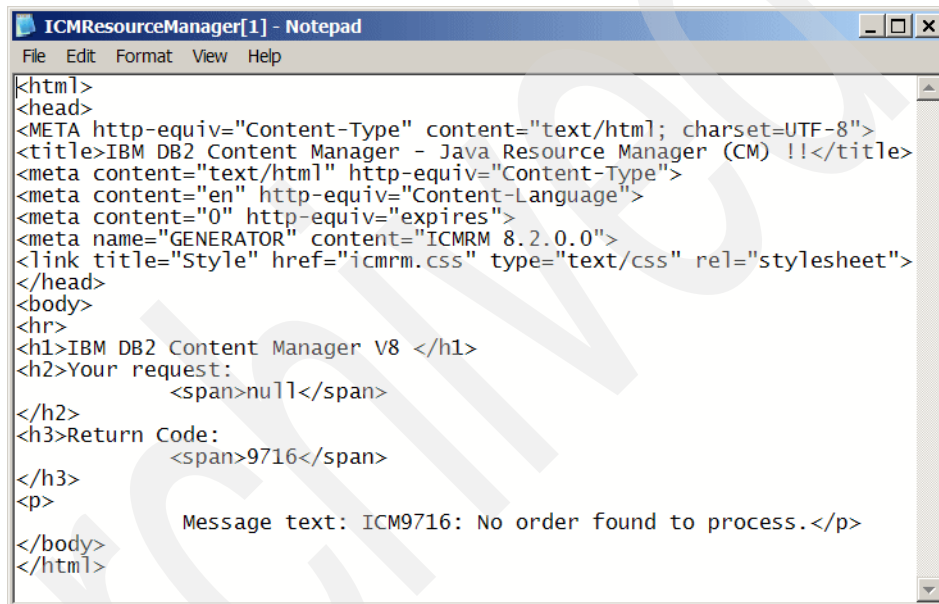


Figure 9-5 Successful reply from a Resource Manager server on multiplatforms

Notice the slightly different connection string to be used when accessing the Resource Manager on multiplatforms: An extra level is added to the path so the full string reads:

`http://ip-address:port-number/icrm/ICMRMResourceManager`

in Figure 9-6, we used the browser option **View** → **Source** to see the content of the reply returned by the Resource Manager. You can see the reply contains information from “IBM Content Manager - Java Resource Manager”; so this reply gives a more certain knowledge that we actually communicated with a Resource Manager.



```
<html>
<head>
<META http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>IBM DB2 Content Manager - Java Resource Manager (CM) !!</title>
<meta content="text/html" http-equiv="Content-Type">
<meta content="en" http-equiv="Content-Language">
<meta content="0" http-equiv="expires">
<meta name="GENERATOR" content="ICMRM 8.2.0.0">
<link title="Style" href="icrmr.css" type="text/css" rel="stylesheet">
</head>
<body>
<hr>
<h1>IBM DB2 Content Manager V8 </h1>
<h2>Your request:
      <span>null</span>
</h2>
<h3>Return Code:
      <span>9716</span>
</h3>
<p>
      Message text: ICM9716: No order found to process.</p>
</body>
</html>
```

Figure 9-6 View the source of the reply from Resource Manager

### 9.12.3 Define z/OS Resource Manager (RM) to multiplatforms RM

In order to implement migration of objects from Resource Manager on Multiplatforms to the “remote” Resource Manager on z/OS, (it is remote from the multiplatforms server’s point of view), we need to identify it to Resource Manager on Multiplatforms.

You can do this by creating a reference to z/OS Resource Manager in the Multiplatforms Resource Manager database using the Content Manager System Administrator Client:

1. Log on as an administrator, open the list of Resource Managers, and expand the definitions for the Multiplatforms Resource Manager.

In our scenario, our Resource Manager is RMPALM.

2. Click **Server definitions** to show the list of servers currently known to the RMPALM server.

As you can see in figure Figure 9-7, the RMPALM server currently knows about itself and the Library Server ITS08LS to which it is connected.

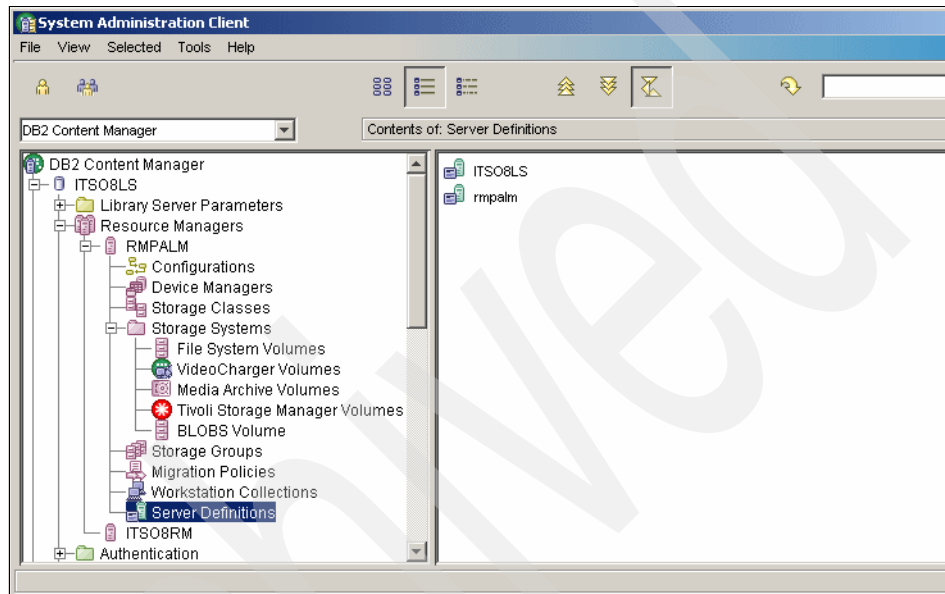


Figure 9-7 List of servers initially known to RM on multiplatforms, RMPALM

3. Right-click **Server Definitions** and select **Add new** from the context menu. This brings you to the screen shown in Figure 9-8.

**New Server Definition**

Name: \* ITS08RM

Server type: Resource Manager

Hostname: \* 9.30.128.228

Platform: \* OS/390

User ID: \* usr009

Password: \* \*\*\*\*\*

Protocol: http

Port number: \* 8094

Schema: \* ITS08ADM

Path: \* /ICMResourceManager

OK Cancel Apply Help

Figure 9-8 Define z/OS RM to the RM on multiplatforms

4. Enter the information for your V8.3 Resource Manager on z/OS:
  - The port number is the port on which your HTTP server on z/OS listens.
  - The Path is the *case-sensitive* string you defined in the httpd.config file.
5. When you have entered all the information, click **OK**.

#### 9.12.4 Define migration rules

When the Resource Manager on multiplatforms stores an object, the object is registered in the RMOBJECTS table, and a range of attributes are set based on the definitions in the collection specified in the store request.

The management class assigned to the collection defines the migration policy of the object. For example, the migration policy specifies how long the object resides on a given media (the storage group) before the Resource Manager should look at moving it. The object registration thus contains the ID of the management class and the date where the object is eligible for processing next time. This processing is done by the Resource Manager migrator task. At that time, the definition of the management class decides what happens to the object next.

In order for the migrator to migrate the objects from your Resource Manager on Multiplatforms to your Resource Manager on z/OS, you need to add and change



a few definitions on your Resource Manager on Multiplatforms using the system administration client.

Specifically, you need to:

1. Define a new storage class on the remote z/OS system to which the objects should be migrated.
2. Update the migration policy for the objects already stored on the Resource Manager on multiplatforms as defined by the management class assigned to the objects.

### Define a new storage class on remote destination z/OS

To define new storage class on remote destination z/OS, perform the following tasks:

1. In the system administration client, under the definitions for your Resource Manager on multiplatforms, select **Storage Classes**. See Figure 9-9.

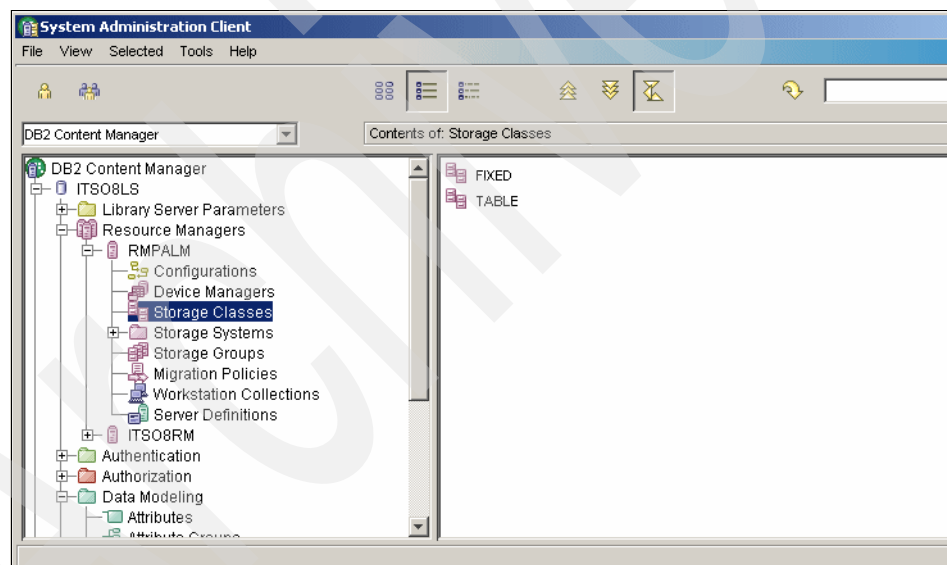


Figure 9-9 Work with storage classes for RM on multiplatforms

2. Right-click **Storage Classes** and select **New** from the context menu.  
You should now see Figure 9-10.

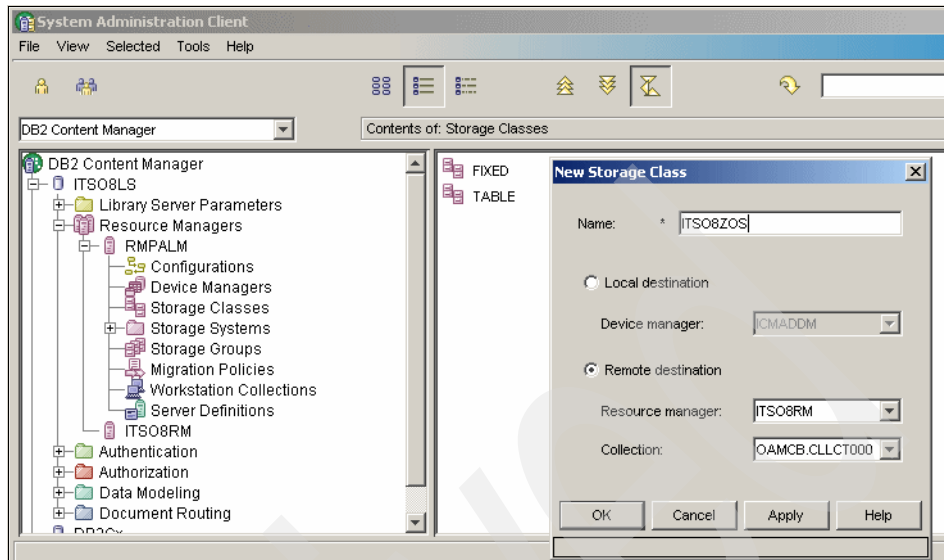


Figure 9-10 Enter details for the new storage class on remote destination on z/OS

3. Define the new storage class as follows:
  - a. In our scenario, we name the new storage class ITSO8ZOS. We will refer to this name later when updating the Migration Policy for the Management Class.
  - b. Select **Remote Destination**. This will define the storage class to be on the Resource Manager on z/OS.
  - c. Select the correct OAM collection on your z/OS system.
  - d. Click **OK** to save the new definition.

When you have completed and saved the new Storage Class definition, you should see the new storage classes on the left panel. See Figure 9-11.

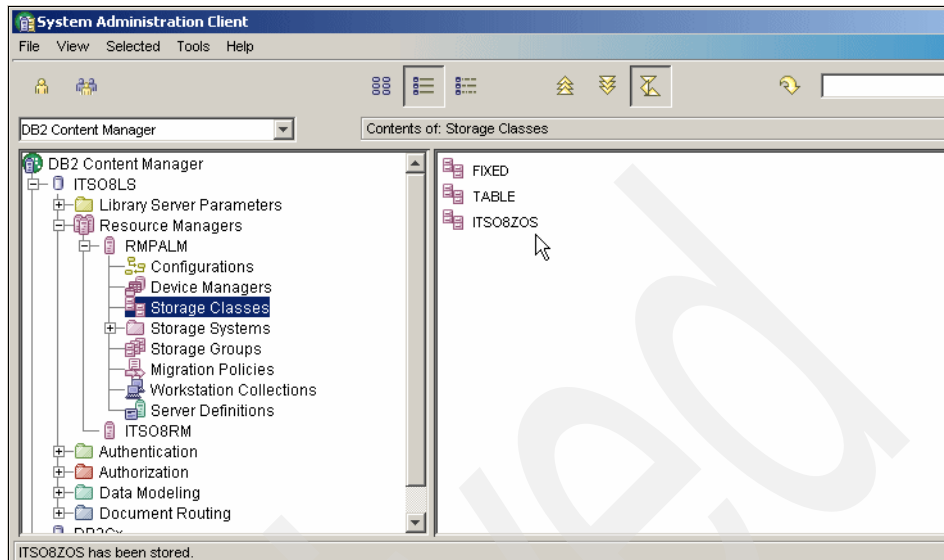


Figure 9-11 New ITS08ZOS storage class has been created

## Update the management class

The objects stored in Resource Manager are registered in the DB2 tables RMOBjects. Each object is assigned a management class when it is stored. The management class defines how the objects are moved between storage groups through its life cycle.

In our scenario, our current management class defines the objects to be kept on the Resource Manager's file system forever. We need to update this definition to define the Content Manager V8.3 Resource Manager on z/OS as the next destination to which to migrate the objects.

Perform the following steps:

1. As shown in Figure 9-12, there is a list of the current management classes of the Resource Manager on multiplatforms on the right panel. Right-click on the one we are going to update and select Properties.

The migration policy properties screen appears as shown in Figure 9-13.

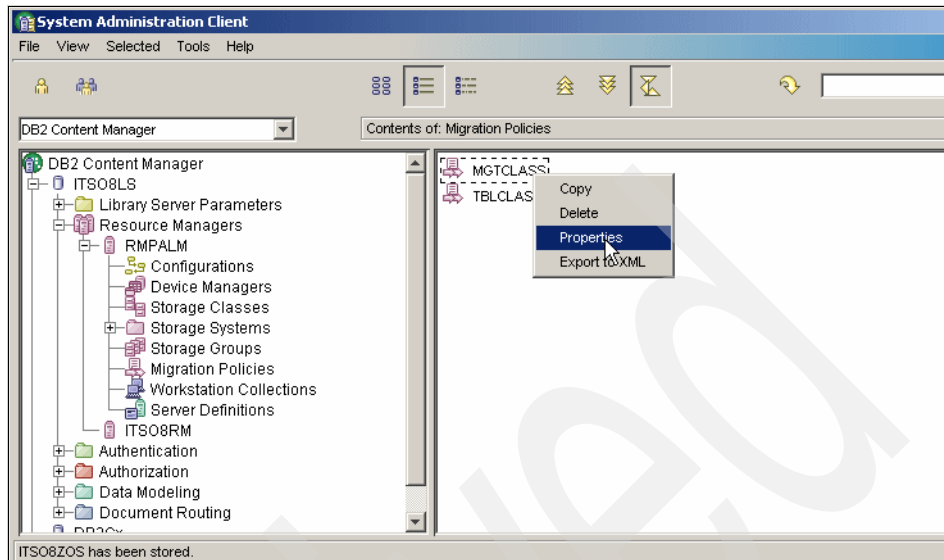


Figure 9-12 Select the management class and show its properties

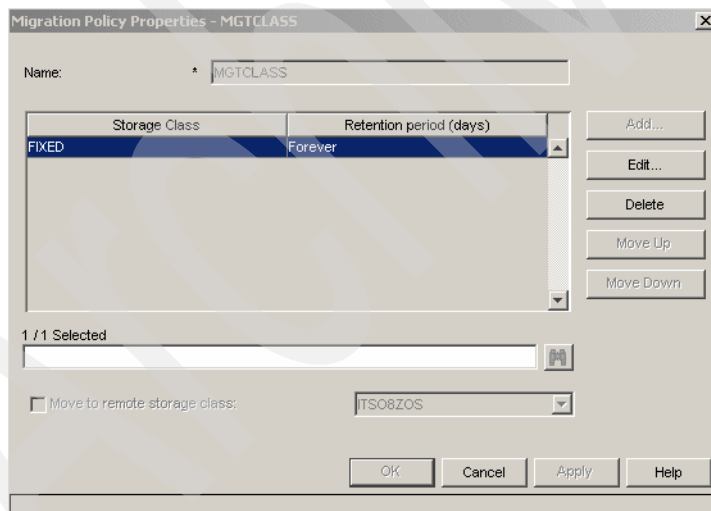


Figure 9-13 Update the migration policy

2. Select **FIXED** entry and click **Edit**.
3. In Figure 9-14, the screen on the left shows the current definition of our management class. Change the retention period from forever to 1 day as shown on the figure on the right-hand side. Click **OK** to save the changes.

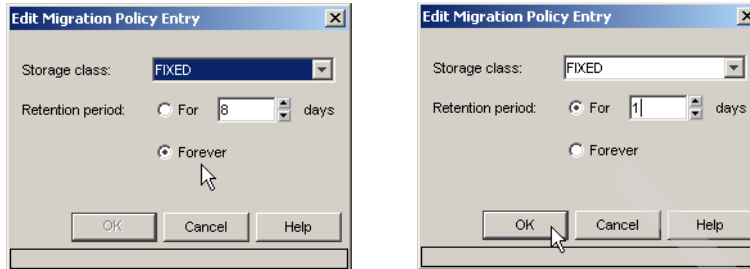


Figure 9-14 Change the retention period from Forever to something less

In Figure 9-15, you can see that now FIXED entry has a retention period of 1 day. In addition, both the Add button and the “Move to remote...” check box are activated.

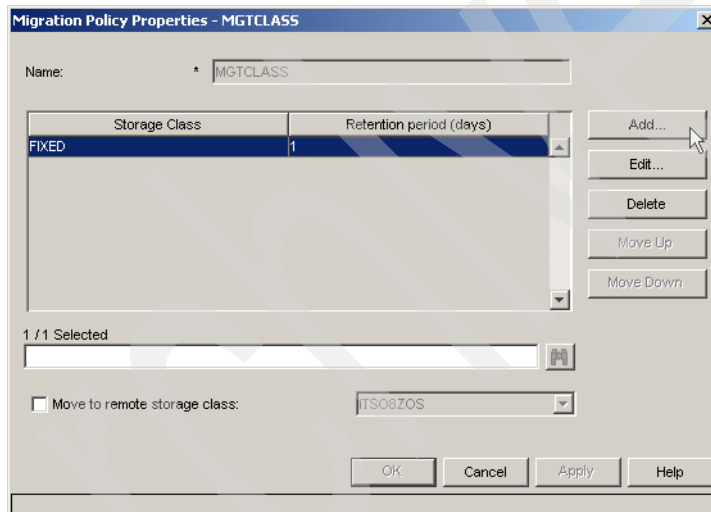


Figure 9-15 Updated migration policy information

You may also notice that the OK button is now greyed out.

In fact, we cannot save a migration policy unless the last entry is marked as Forever or it is moved to a remote storage class.

This is due to the fact that the storage management system would not know what to do with the object once the retention period of the last entry is reached if there is no next step. So there must be a final Forever entry or a remote location. In the latter case, it will then be the remote Resource Manager that controls the subsequent migration policies for the objects.

4. Check the **Move to remote storage class** check box to activate the migration of objects for this management class.
5. Select the **ITS08ZOS** storage class we defined in, “Define a new storage class on remote destination z/OS” on page 267. See Figure 9-16.
6. Click **OK** to save the update.

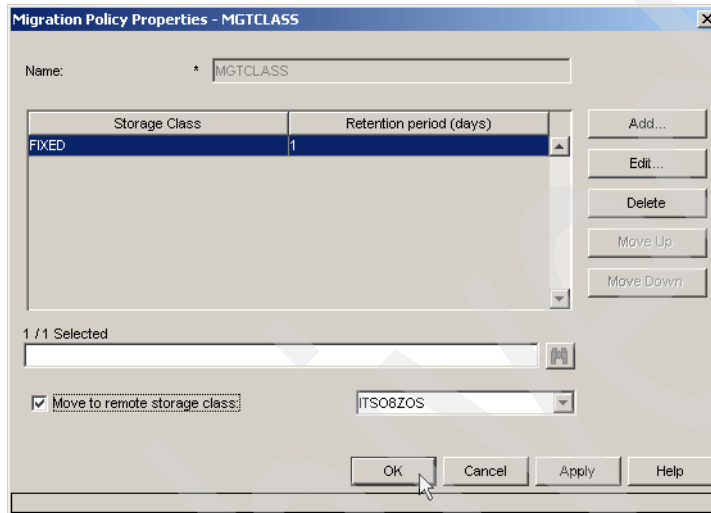


Figure 9-16 Select the target remote storage class

### 9.12.5 Status after updates

We have now completed the changes to the migration policy of the management class.

These changes will only take affect in the following situations:

- ▶ When new objects are stored in collections using the updated management class. Because it is at the time of storing the object that the information about the management class' migration policy is read from the Resource Manager definition table. And at storage time, the value for OBJ\_ACTIONDATE is calculated and saved in the object registration in the Resource Manager table RMOBJECTS. In addition, the object's registration will contain a reference to the management class assigned to the object.
- ▶ When objects that due to their current OBJ\_ACTIONDATE are processed by the migrator task. The migrator task will at this time read the current definition for the Management Class and process the object according to the updated definition.

This indicates that objects currently registered in the RMOBJECTS table as having reached the location marked with Forever in the migration policy (with an action date of 31st December 9999) will not be processed by the migrator task and therefore they will not be migrated before we make them eligible for migration. We address this issue in 9.12.7, “Start migration of objects” on page 273.

## 9.12.6 Setup migrator task

The migrator task runs as a service on the Content Manager V8.3 Resource Manager server. The task processes objects based on a schedule defined through the system administration client.

We set the migrator to run every day. You can decide to have the migrator run outside of normal business hours to defer the load on the system.

## 9.12.7 Start migration of objects

In this section, we discuss how you can start the migration of objects.

### How many items are ready for migration?

Based on the value of the OBJ\_ACTIONDATE, the migrator task processes those objects with a date equal to or less than today's date.

To get the number of items the migrator may process and also the amount of data on which the migrator needs to be working, you can use the SQL shown in Example 9-17.

*Example 9-17 Determine number of items eligible for processing by migrator*

---

```
--
-- Create count on items eligible to be processed by migrator task
--
-- Set schema name for the Resource Manager tables
set current SQLID = 'RMADMIN';

-- Number of objects grouped by date
select count(*), obj_actiondate
from rmoobjects
where obj_actiondate <= current date
group by obj_actiondate
;

-- Number of objects and amount of data
select count(*) as CNT, sum(obj_size) as SIZE
from rmoobjects
```

```
where obj_actiondate <= current date
group by obj_actiondate
;
```

---

## Mark items for migration

To force the objects to be available for migration, we must update the OBJ\_ACTIONDATE for the objects we want the migrator to process.

Set the date of these objects to be today's date so that the migrator will process them the next time the RMOBJECTS table is processed by the migrator.

The SQL in Example 9-18 illustrates how to update the OBJ\_ACTIONDATE value.

### *Example 9-18 Update the OBJ\_ACTIONDATE value*

---

```
--
-- Update OBJ_ActionDate so objects will be processed by migrator task
--
-- Set schema name for the Resource Manager tables

Set current SQLID = 'RMADMIN';
--
Update RMOBJECTS
Set OBJ_ActionDate = current date
where select_criteria
;
```

---

With this approach, we recommend you specify a selection criteria in order to limit the number of rows you update; otherwise, without a select criteria, all objects will be available for the migrator to process.

We recommend that you migrate the objects in smaller groups. Work out a criteria on which to select the objects. These criteria could be:

- ▶ Date the object was stored.
- ▶ Management Class of the object.
- ▶ First *nn* characters of the ItemID. Based on a previous "Select COUNT(\*) where ItemID like 'xxx%' ", you could control how many items would be updated and processed.

The item type or component type the object belongs to is not available on the Resource Manager. If you would like to migrate by a criteria such as item type, you would need to build some more complex selection criteria based on a JOIN with a copy of some of the Library Server tables.



Archived





## Part 3

# Logs and advanced configuration topics

In this part, we cover how to activate traces and set up logs to maintain and troubleshoot your Content Manager for z/OS system.

We discuss:

- ▶ Advanced configuration topics
- ▶ Configuration of multiple Content Manager instances in the same LPAR
- ▶ Issues and considerations when setting up a Content Manager system in a sysplex environment
- ▶ Changing default access to DB2 plans and packages.

In the end of the last chapter, we also look at error situations you may encounter and where to find information to help you analyze these situations.



## Traces and logs

This chapter provides information on how to activate traces, and where to set up and obtain the necessary logs for debugging your Content Manager for z/OS system.

We cover the following topics:

- ▶ Library Server traces and logs
- ▶ Resource Manager traces and logs
- ▶ Client's traces and logs

## 10.1 Introduction

Content Manager for z/OS provides logs and traces that help you efficiently and properly diagnose problems and issues that have occurred in your system.

A Content Manager for z/OS system contains three main components: Library Server, Resource Manager, and clients. It is important for you to be familiar with these components so that it will be easier for you to isolate problems and resolve problems when they do occur.

The following tips help you get started isolating problems in your system:

- ▶ If you have a problem searching a document, the problem is probably related to a Library Server issue.
- ▶ If you are able to search a document, but you are not able to retrieve the document to display it on a screen, the problem may be related to a Resource Manager issue.
- ▶ If the client application is getting other types of issues, it may be client, Library Server, or Resource Manager issues.
- ▶ If you cannot import a document, it may be a problem with Library Server, Resource Manager, or the client application.

Logs and traces are provided for Library Server, Resource Manager, and client components. Because the traces and logs increase the system workload, we recommend turning on the traces only for debugging purposes.

## 10.2 Library Server traces and logs

The Library Server is the component responsible for storing and retrieving content metadata. The Library Server also manages and controls access for a Content Manager system.

### 10.2.1 Enabling trace for Library Server

To enable the trace for Library Server, use the system administration client and execute the following steps:

1. Launch the system administration client.
2. Expand your Library Server name and select **Library Server Parameters**.
3. Right-click **Configuration** and select **Explore**.
4. Right-click **Library Server Configuration** and select Properties.

5. In the Library Server Configuration screen, select the **Log and Trace** tab. See Figure 10-1.

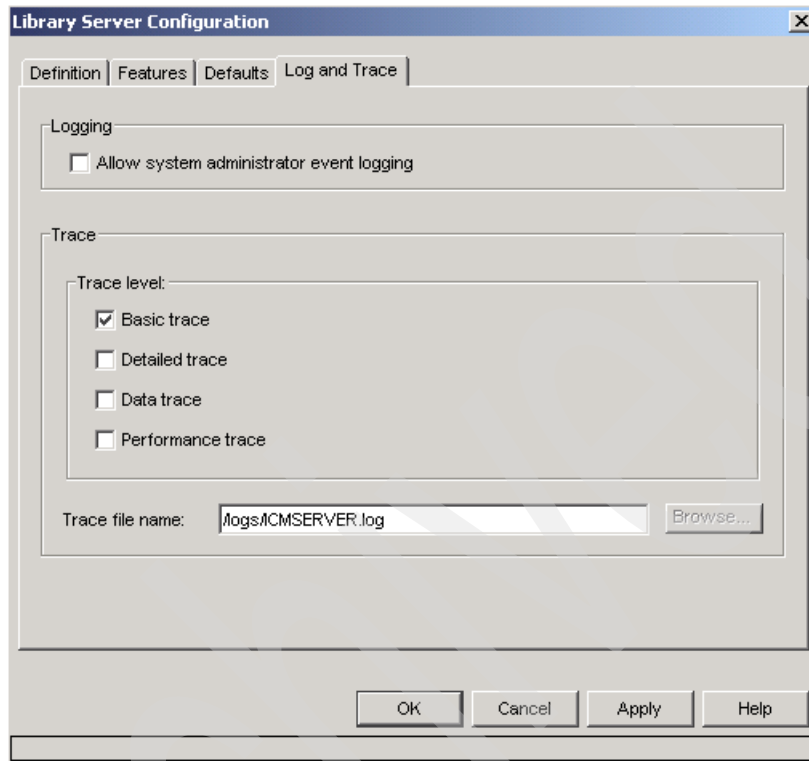


Figure 10-1 Library Server Configuration: Log and trace tab

6. Select the trace level that you want to enable and specify the trace file to write the trace. The initial value for the trace file name was loaded during the installation of Content Manager Library Server. For Content Manager z/OS, the default value is the DD card SYSPRINT in the WLM procedure. Alternatively, you can specify a file in the z/OS UNIX System Services (USS) directory.
7. Click **OK** to exit the Library Server Configuration screen and save the changes.

When we set the trace level, it updates the column TRACELEVEL from the Library Server table ICMSTSYSCONTROL with numeric values. Each value indicates a trace level. See Table 10-1.

Table 10-1 Library Server trace levels (set in ICMSTSYSCONTROL table)

Trace value	Description
0	No trace
1	Basic trace
2	Detailed trace
4	Data trace
8	Performance trace
16	Build and parse
32	Memory management
256	Cache trace
512	Cache allocation
1024	Cache management
3	Basic and detailed trace
5	Basic and data trace
9	Basic and performance trace
15	Basic, detailed, data, and performance trace
63	Basic, detailed, data, performance, build/parse, and memory management trace

From Table 10-1, we see that the sum of the trace values results in a combination of a trace level. The basic, detailed, data, and performance traces can be set using the system administration client GUI interface. The other traces can only be set through the DB2 update statement.

To confirm the actual trace level, you can issue the following DB2 query:

```
select tracelevel from <creator>.icmstsyscontrol;
```

To update the tracelevel value:

```
update <creator>.icmstsyscontrol set tracelevel = <value>;
```



**Note:** In Content Manager V8.2, the positive values for the TRACELEVEL logged only information regarding the client. If you wanted the system administration client actions logged also, you need to set a negative value.

In Content Manager V8.3, setting either a negative or positive value will log for both the client and system administration activities.

The output log contains information similar to Figure 10-2.

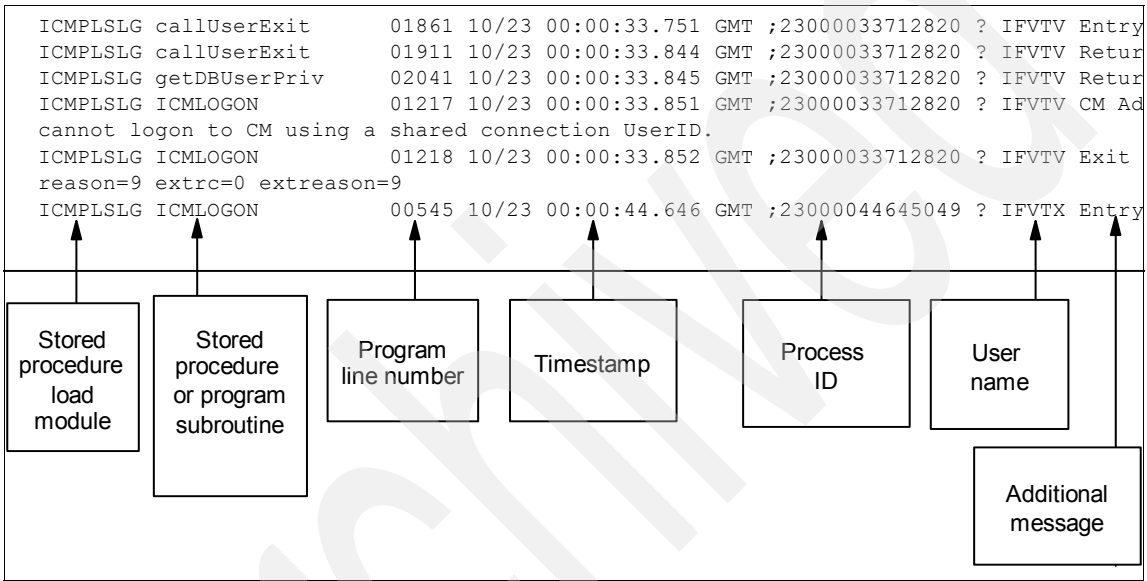


Figure 10-2 Log file output

Use these traces for debugging purposes only. Depending on what trace level you set, it may impact the overall Content Manager system performance. For best performance, the TRACELEVEL value should be 0.

### 10.2.2 DB2 joblogs

You can look for additional messages regarding DB2 in the xxxxMSTR address space, where xxxx is the DB2 subsystem name.

## 10.3 Resource Manager traces and logs

Resource Manager is responsible for storing and retrieving documents.

There are three distinct types of traces for Resource Manager:

- ▶ Resource Manager trace
- ▶ Resource Manager DB2 trace
- ▶ HTTP trace

**Note:** There is no system administration client GUI interface to enable these traces. We must set the trace manually.

### 10.3.1 Resource Manager trace

Resource Manager has its own control table called ICMRMCONTROL where we can set the TRACELEVEL. The possible trace level values are described in Table 10-2.

Table 10-2 Resource Manager trace levels (set in ICMRMCONTROL table)

Trace value	Description
0	No trace
1	Basic trace
2	Detailed trace
3	Verbose

The DB2 trace, or DSNTRACE, contains all interactions between the Resource Manager and DB2. The output of this trace will be in DSNTRACE output of the Webserver process.

To confirm the actual trace level, you can issue the following DB2 query:

```
select tracelevel from <creator>.icmrcontrol;
```

To update the tracelevel value:

```
update <creator>.icmrcontrol set tracelevel = <value>;
```

**Note:** In our tests, if you specify a TRACELEVEL value greater than 3 in table ICMRMCONTROL, Resource Manager assumes value 3 in the output trace.

### 10.3.2 Resource Manager DB2 trace

To trace Resource Manager problems related to DB2, we need to set the following DD statements in the HTTP Server procedure:

```
//STDERR DD SYSOUT=*,OUTPUT
//SYSOUT DD SYSOUT=*,OUTPUT
//CEEDUMP DD SYSOUT=*,OUTPUT
//DSNTRACE DD SYSOUT=*
```

### 10.3.3 HTTP server trace

To set the verbose trace on in the HTTP server, you need to add -vv option in the parameter ICMSPARM from the HTTP Server procedure. For example:

```
ICSPARM='-vv -p 2030 -r /etc/httpd.conf.2030'
```

The output trace is generated in //SYSOUT DD card.

Because the resulting output could be very large, we recommend using this setting only during troubleshooting.

## 10.4 Client's traces and logs

In this section, we discuss the client logs in a Windows environment.

### 10.4.1 Client for Windows

In the Client for Windows, there are two fields regarding the logs:

- Log configuration directory

It indicates the location for the file ICMClientLog.ini. The default directory is %IBMCMROOT%\CMGMT\icmclient\<username>. The configuration file ICMClientLog.ini contains a column with a list of log options and another column with the letters “d” (disable) or “e” (enable).

Example 10-1 shows a portion of ICMClientLog.ini content.

*Example 10-1 Portion of ICMClientLog.ini content*

---

; e=enabled, d=disabled	
applicat	d
asfbuild	d
asform	e
asparfrm	e
basicsca	e
basicsrc	d
contitem	e
docts	e
docview	d
docwin	e

For debugging, we need to enable all the options (“e”).

► Log output directory:

This is the location where the log is written. The default directory is %IBMCMROOT%\CMGMTLOG\icmclient\

- ICMClient.err: Contains only error messages.
- ICMClient.log: Contains the client trace log and the error messages.

Note, if the trace is enabled, ICMClient.log contains the trace information. If the trace is not enabled, it contains the error messages just as ICMClient.err file.

To confirm the log directories, in the Client for Windows, select from the menu, **Options** → **Preferences**. This opens the preference screen as shown in Figure 10-3.

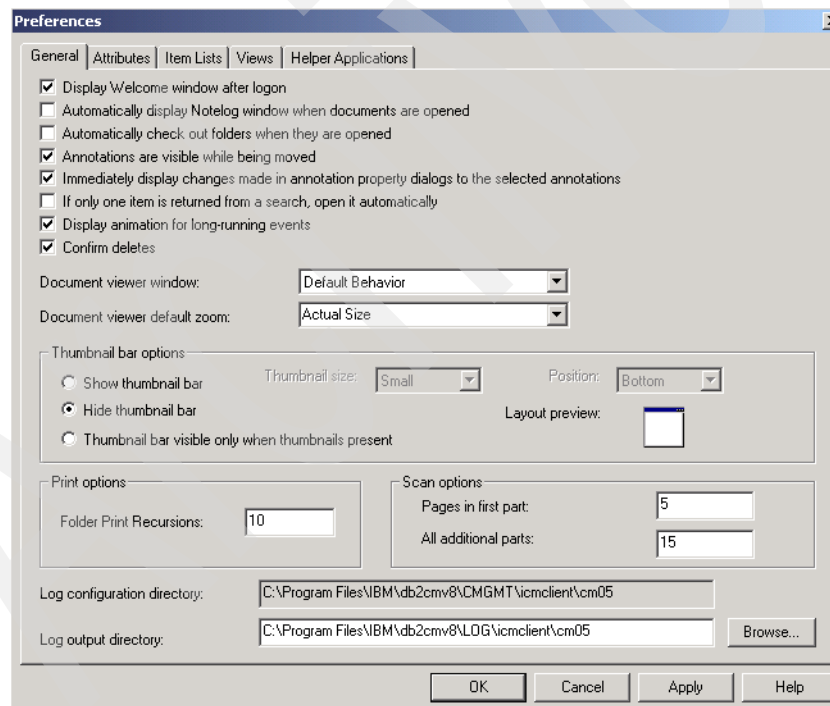


Figure 10-3 Client for Windows log preference setting

**Note:** This log also applies for applications that use the Content Manager Client for Windows OLE APIs.

## 10.4.2 System administration client

Within the Content Manager system administration client, there is a log configuration utility where you can configure logs for different components.

### Log Configuration Utility

The log configuration utility is new for Content Manager V8.3. You can get there by launching the system administration client, and from the menu, select **Tools** → **Log Configuration**. Figure 10-4 appears.

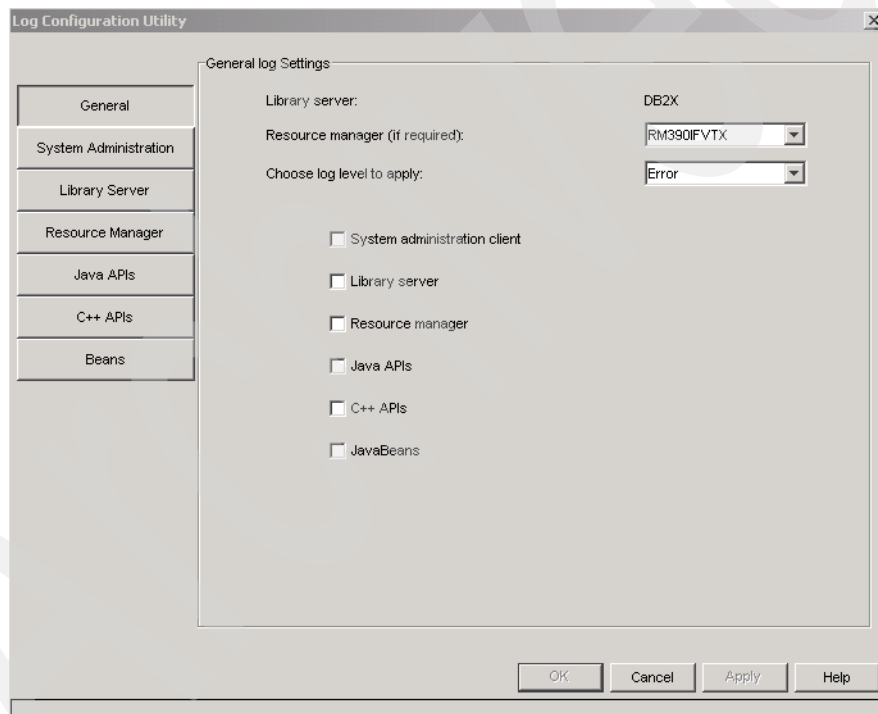


Figure 10-4 System administration client: Log configuration utility

With the log configuration utility, we can configure the logs for:

- ▶ System administration client
- ▶ Library Server
- ▶ Resource Manager (it is not valid for Resource Manager in z/OS)
- ▶ APIs (Java)

- ▶ APIs (C++)
- ▶ Beans

For more information about the Log Configuration utility, refer to the manual, *IBM DB2 Content Manager for z/OS V8.3: System Administration Guide*, SC27-1335, Chapter 10, “Managing servers in DB2 Content Manager, section Logging and Trace utility”.

## Event logging

Content Manager is able to log two types of events for audit purposes:

- ▶ System administration events
- ▶ Item events

### ***System administration events***

The system administration events are actions performed by an administrator using either the system administration client or a custom application. Examples of these events include defining users, assigning privileges, and assigning an access control list to an object. These events are stored in table ICMSYSADMEVENTS.

You can choose to enable the logging of system administration events to keep a record of the changes performed within any System Administration Client. For example, you can enable system administration events to record the event in the log whenever you define a user, assign privileges, or perform other system administration tasks. This provides an audit trail, where the administrator can get a history of the administration events from the ICMSYSADMEVENTS table when needed.

To enable logging, we need to use the system administration client and execute the following steps:

1. Launch the system administration client.
2. Expand your Library Server name and click **Library Server Parameters**.
3. Right-click **Configuration** and select **Explore**.
4. Right-click **Library Server Configuration** and select Properties.
5. In the Library Server Configuration screen, click the **Log and Trace** tab.
6. Check the Allow system administrator event logging check box as shown in Figure 10-5.

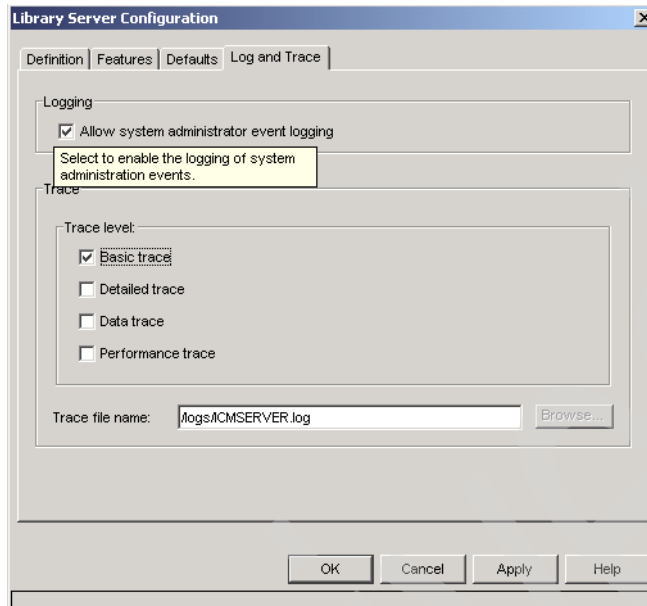


Figure 10-5 System administration client: System administration event logging

### Item events

Item events are actions performed against specific objects within Resource Manager, or the object indexing information within Library Server. These events are stored in the table ICMSTITEMEVENTS.

You can choose to enable item events to keep a record of the events that happened to an item. For example, you can enable read events to write to the event log whenever a document is retrieved for display or printing. Or, you might want to enable create events so that you have a record when the document or folder is created.

To enable the item events logging, modify the item type you wish to log using the system administration client:

1. Expand your item types and choose the item type you wish to log.
2. Right-click **Properties**.
3. In the item type properties screen, select the **Logging** tab.
4. Select the options you want to log and click **OK** (see Figure 10-6).

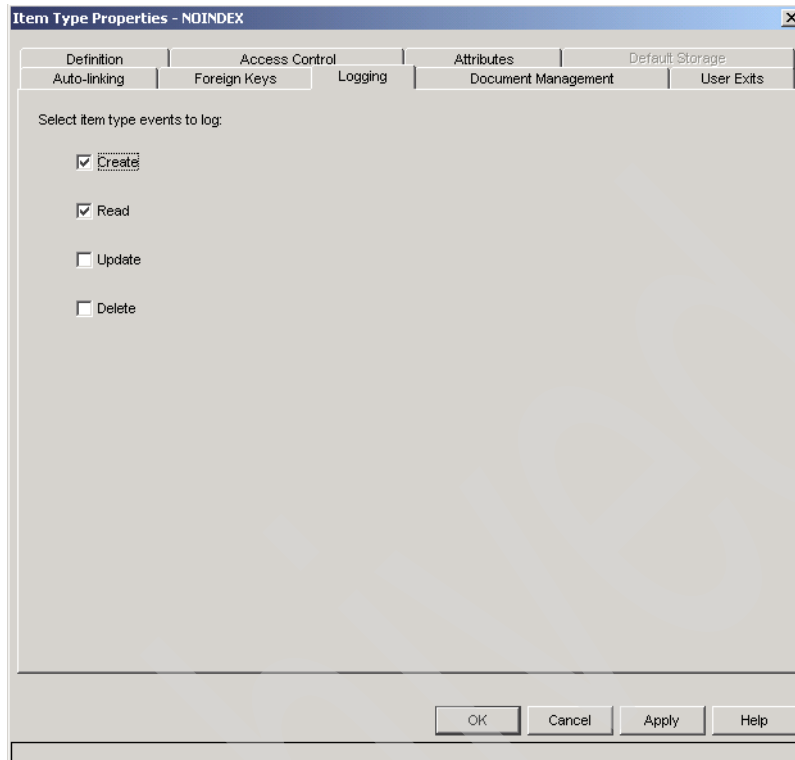


Figure 10-6 System administration client: Enabling item events logging

As both system administration events and item events are stored in DB2 tables, we can issue select statements to browse the data.

**Important:** Make sure that these tables do not reach their storage capacity; otherwise, attempts to log events may fail, and the entire action will rolled back.

For more information about system administration events, item events, and how to maintain the ICMSTSYSADMEVENTS/ICMSTITEMEVENTS tables, consult the manual, *IBM DB2 Content Manager for z/OS V8.3: System Administration Guide*, SC27-1335, Chapter 10, “Managing servers in DB2 Content Manager, section Logging and Trace utility”.

### 10.4.3 UDB traces and logs

Sometimes, the Content Manager traces are not enough to debug the issue. You need additional traces and logs to identify and isolate the problem. As Content



Manager information resides in DB2 tables and the client applications access that information, you can set DB2 traces in the workstation where the application runs. This applies for the system administration client, Client for Windows, and user applications.

Since this kind of low level tracing can be detrimental to performance, we recommend that you only enable this tracing when you have a request to do so by a support representative.

## DB2DRDAT or DDCSTRC

The db2drdat (also known as ddcstrc) allows a user to capture the DRDA (Distributed Relational Database Architecture™) data stream exchanged between a DRDA application requester and DB2 DRDA application server. It applies to Content Manager for z/OS.

To capture the trace:

1. From a DB2 command window, type:

```
db2drdat on
```

It will display the following message:

```
Trace is turned on
```

2. Start the client application.
3. Get to the point where you are having problems, stop the trace, and specify a name for the trace (-t option) as follows:

```
db2drdat off -t=<tracefile>
```

It will display the following message:

```
Trace truncated           : NO
Trace wrapped             : NO
Total number of trace records : 180
Number of trace records formatted : 180
Trace is turned off
```

After the trace is generated, you can open it with a text editor to examine where the system has failed. For more information about this command, refer to the manual, *DB2 Universal Database Command Reference V8*, SC09-4828.

**Note:** As db2drdat captures the DRDA information, this command is not available in the DB2 Runtime Client. If the client application is using the DB2 Runtime Client, we must set the trace in the connection gateway, such as DB2 Connect.

## DB2TRC

The db2trc records information about DB2 operations and formats this information in a readable format.

To capture the trace, use the following procedure:

1. From a DB2 command window, type:

```
db2trc on
```

It will display the following message:

```
Trace is turned on
```

2. Start the client application.
3. Get to the point where you are having problems, dump the trace information to a file as follows:

```
db2trc dmp <filename>
```

4. Stop the trace.
5. The generated trace is not readable, you need to format it with the following command:

```
db2trc fmd <filename> <target_filename>
```

It displays the following message:

```
Trace truncated           : NO
Trace wrapped             : YES
Total number of trace records : 11345
Number of trace records formatted : 11345
```

After the trace is generated, you can open it with a text editor. For more information about this command, refer to the manual, *DB2 Universal Database Command Reference V8*, SC09-4828.

## CLI trace

DB2 CLI (Call Level Interface) is the IBM callable SQL interface to the DB2 family of database servers. The Content Manager client applications use this interface.

To capture a CLI trace, use the following procedure:

1. Edit the file db2cli.ini located at \sqllib path from the Windows workstation.

Include the following lines in the section [COMMON].

```
[COMMON]
Trace=1
TracePathName=c:\temp\cli\trace
TraceComm=1
TraceFlush=1
```

```
TraceTimestamp=1
```

2. Save the file.
3. Start the application.
4. Get to the point where you are having problems, set Trace=0 in db2cli.ini. It disables the trace generation.

The trace is generated in the directory specified in the *TracePathName* parameter. It generates a file with .cli extension.

The *TraceFlush* parameter specifies how often trace information is written to the CLI trace file. The value 1 means it writes the information to disk at each CLI trace entry. This keyword has a great impact on the client application performance, but it ensures all the trace data was written successfully.

For more information about this trace, refer to the manual, *DB2 Universal Database Call Level Interface Guide and Reference, Volume 1*, SC09-4849, Chapter 20. “CLI/ODBC/JDBC trace facility”.

## JDBC trace

There are some client applications (such as the system administration client) that access Content Manager z/OS through JDBC.

To capture this trace, use the following procedure:

1. Edit the file db2cli.ini located at \sqllib path from the Windows workstation.

Include the following lines in the section [COMMON].

```
[COMMON]
JDBCTrace=1
JDBCTracePathName=c:\temp\jdbcctrace
JDBCTraceFlush=1
```

2. Save the file.
3. Start the application.
4. Get to the point where you are having problems, set JDBCTrace=0 in db2cli.ini. This disables the trace generation.

The trace is generated in the directory specified in the *JDBCTracePathName* parameter. The trace file has the .trc extension.

The JDBCTraceFlush=1 has a great impact on the client application performance, but it ensures that all the trace data is written to the file when the application terminates abnormally.

For more information about this trace, refer to the manual, *DB2 Universal Database Call Level Interface Guide and Reference, Volume 1*, SC09-4849, Chapter 20, “CLI/ODBC/JDBC trace facility”.

#### 10.4.4 Content Manager Toolkit for z/OS traces and logs

The Content Manager Toolkit for z/OS has the file *cmblogconfig.properties* located at the USS directory %IBMCMROOT/cmgmt/connectors for tracing purposes. This file is divided into multiple sections. Within each section, there is a list of parameter keys, their valid values, and explanations. We recommend reading them carefully before setting any logs and traces.

## Multiple Content Manager instances in the same LPAR

In this chapter, we discuss how to configure multiple Library Servers and Resource Managers in the same LPAR. We also cover how to configure the clients to connect to each system. This is useful if you need to have two or more Content Manager test environments but you do not want to put workload on another z/OS partition.

We cover the following topics:

- ▶ Multiple Library Servers in the same DB2 subsystem
- ▶ Multiple Resource Managers in the same DB2 subsystem
- ▶ Configure Content Manager clients
- ▶ XML export

## 11.1 Introduction

When planning for a Content Manager installation, there may be a requirement to have multiple environments running at the same time. The requirements may differ among installations, but in most cases, you have a need for environments for development, test, education, and production.

You may choose to isolate the environments by installing them in separate LPARs. This approach is the simplest and most straightforward approach from an installation and maintenance point of view.

In other cases, there may be a need to have more than one instance of Content Manager running in one LPAR. In these scenarios, it is a good idea to keep related instances in the same LPAR such as the test and development environment.

When installing multiple instances of the Library Server and Resource Manager in the same LPAR, you need to implement a proper naming standard for the components including the following:

- ▶ Library Server (including token values)
- ▶ Resource Manager (including token values)
- ▶ OAM Collections/Storage Group
- ▶ HTTP server
- ▶ Workload Manager (WLM) environment

Each instance would have its own set of users, user groups, executable libraries, databases, HTTP servers, and WLM environment.

**Important:** When planning for multiple environments, remember that you *cannot* share one Resource Manager between multiple instances of a Library Server, whether this is in the same or different LPARs. You may be tempted to “reuse” an existing Resource Manager when testing a new instance of a Library Server, but we strongly discourage you from trying this. It will not work.

This chapter complements Chapter 6, “Installation” on page 153. The recommendations provided in that chapter apply to this chapter as well.

## 11.2 Multiple Library Servers in the same DB2 subsystem

When we connect to a Library Server, we access Content Manager DB2 tables that are associated with a specific Library Server schema name and database

name. To accommodate different instances in the same LPAR, we need to configure them in the installation jobs and make the clients (both the system administration client and the Windows client) recognize the different systems.

Make sure they do not share the OAM collection. Using separate collections helps isolate the object captures, so you can manage the environments more easily.

If you intend to install two or more Library Servers in the same DB2 subsystem, we list the variables that *must* be unique for each Library Server.

**Note:** In DB2 z/OS, the database name is just a logical name. The only way to distinguish each Library Server's DB2 objects is changing the variables.

We also include additional comments if you want to install Library Server in different DB2 subsystems in the same LPAR.

We recommend copying the jobs from ?ICM?.SICMINS1 to another data set and customizing the jobs in the new data set.

### Pre-allocate data sets

We must allocate data sets <HLQ>.ICMUSQL and <HLQ>.PROCLIB for each new Content Manager instance. This prevents a Content Manager system overriding some jobs or procedures in another Content Manager system.

**Note:** When installing Library Server in another DB2 subsystem in the same LPAR, allocating separate data sets <HLQ>.ICMUSQL and <HLQ>.PROCLIB for each new Content Manager instance applies for *every* installation.

### ICMMLSCR

This job creates the DB2 objects in Library Server.

If you want to configure different Content Manager instances in the same DB2 subsystem, the following variables must be different for each Library Server configuration:

- ▶ ?DB2PKGCOLLID? - Collection name of the DB2 package
- ▶ ?CREATOR? - ID of creator defining the DB2 objects
- ▶ ?DATABASE? - Database name for Library Server
- ▶ ?WLMENV? - Workload manager environment in which Content Manager stored procedures run

**Note:** When installing Library Server in another DB2 subsystem in the same LPAR, you only need to take care of the ?WLMENV? variable. Even if you are installing Content Manager in a different DB2 subsystem, the WLM application environment must have a different name. Refer to Chapter 6, “Installation” on page 153 on how to configure the WLM application environment.

## ICMMLSLD

This job loads some default information into Content Manager Library Server DB2 tables.

If you want to configure different Content Manager instances in the same DB2 subsystem, the following variables must be different for each Library Server configuration:

- ▶ ?CREATOR? - Qualifier of DB2 tables
- ▶ ?DATABASE? - Database name
- ▶ ?RMINSTNAME? - Default Resource Manager instance name
- ▶ ?RMHOSTNAME? - Default Resource Manager host name or IP address
- ▶ ?RMPORTNUM? - Default Resource Manager port number

**Note:** When installing Library Server in *another* DB2 subsystem in the same LPAR, the ?RMINSTNAME?, ?RMPORTNUM? must be different for each Content Manager system.

## ICMMLSBD and ICMMBIND

These jobs bind the Library Server packages and plans.

If you want to configure different Content Manager instances in the same DB2 subsystem, the following variables must be different for each Library Server configuration:

- ▶ ?ACLPLAN? - DB2 plan for ICMPACL
- ▶ ?SAPPLAN? - DB2 plan for ICMPPLSAP
- ▶ ?DB2PKGCOLLID? - Collection name of the DB2 packages
- ▶ ?CREATOR? - ID of creator defining the DB2 objects
- ▶ ?OWNER? - Owner of the DB2 packages

**Note:** When installing Library Server in *another* DB2 subsystem in the same LPAR, you can use the same values in another Library Server database in a second DB2 subsystem.



## ICMMLSGT

This job grants plan execute access for the users.

Before running this job, you need to configure the following variables:

- ▶ ?ACLPLAN? - DB2 plan for ICMPCACL
- ▶ ?SAPPLAN? - DB2 plan for ICMPPLSAP
- ▶ ?DB2PKGCOLLID? - Collection name for stored procedure packages
- ▶ ?CREATOR? - ID of creator defining the DB2 objects

**Note:** When installing Library Server in *another* DB2 subsystem in the same LPAR, you can use the same values in another Library Server database in a second DB2 subsystem.

## ICMMCACL

This job Initializes the compiled ACL table with the user IDs and privileges.

The following variables must be unique for each Content Manager system if it is in the same DB2:

- ▶ ?ACLPLAN? - DB2 plan for ICMPCACL
- ▶ ?CREATOR? - ID of creator defining the DB2 objects
- ▶ ?DATABASE? - Database name

**Note:** When installing Library Server in *another* DB2 subsystem in the same LPAR, you can use the same values in another Library Server database in a second DB2 subsystem.

## WLM application environment and WLM procedure

We must define at least one WLM application environment for each Library Server. The WLM procedure names must also be different. If you have the DD statement //ICMSQL and you intend to use the same Content Manager PROCLIB data set, you need to change the member names (for ICMMLSQx).

**Note:** When installing Library Server in *another* DB2 subsystem in the same LPAR, the information above also applies to this installation.

## ICMMLSQ1 or ICMMLSQ2 or ICMMLSQ3

If you intend to use ICMMLSQ2 or ICMMLSQ3, you need to point the variable ?ICMUSQL? to different PDS names. Otherwise, the generated jobs may be overwritten by another Content Manager system when updating an item type.

By default, Content Manager runs the jobs (ICMMLSQx) with job name ICMUJOB. You can change it to a name specific for your environment.

**Note:** When installing Library Server in another DB2 subsystem in the same LPAR, the information above also applies to this installation.

For more information on ICMMLSQ1, ICMMLSQ2, or ICMMLSQ3, refer to “Step 3: Create user tables” on page 158.

## 11.3 Multiple Resource Managers in the same DB2 subsystem

In the same way that we can have different Library Servers in the same DB2 subsystem, we can also have different Resource Managers in the same DB2 subsystem or LPAR.

Remember that Resource Manager cannot be shared between two or more Library Servers. That is, one Library Server can be connected to one or more Resource Managers, but each Resource Manager can be connected to only one Library Server.

If you intend to install two or more Resource Managers in the same DB2 subsystem, we list the variables that need to be unique for each Resource Manager. This complements Chapter 6, “Installation” on page 153. The recommendations we have for that chapter apply to this section as well.

We recommend copying the jobs from ?ICM?.SICMINS1 to a new data set and customizing the jobs in the new data set.

### ICMMRMCR

This job creates the DB2 objects to Resource Manager.

The following variables need to be configured and must be unique for each Resource Manager:

- ▶ ?CREATOR? - The qualifier of Content Manager database
- ▶ ?STOGROUP? - The name of the DB2 Storage Group
- ▶ ?ICMMRMDB? - DB2 Content Manager Resource Manager database name

**Note:** When installing Resource Manager in another DB2 subsystem in the same LPAR, the variable values may be the same.

## ICMMRMLD

This job loads default definitions for Resource Manager.

The following variables need to be configured and must be unique for each Resource Manager:

- ▶ ?CREATOR? - ID of creator defining the DB2 objects
- ▶ ?RMNAME? - Resource Manager name
- ▶ ?LSDBNAME? - The Library Server database name (should be different only if it does not connect to the same Library Server)

**Note:** When installing Resource Manager in *another* DB2 subsystem in the same LPAR, the variables ?RMNAME? must be different and ?LSDBNAME? should be different if it does not connect to the same Library Server.

## ICMMRMBD or ICMMRMBR

If the Library Server is in the same DB2 subsystem, run job ICMMRMBD to bind the Resource Manager packages/plans. Otherwise, if it is in a remote location, run bind job ICMMRMBR.

The following variables need to be configured and must be unique for each Resource Manager:

- ▶ ?CREATOR? - The qualifier of Resource Manager database
- ▶ ?LSSHEMA? - The Library Server database schema name
- ▶ ?ICMMOPLN? - Plan name for Resource Manager CGI program
- ▶ ?ICMMOSAP? - Plan name for Asynchronous Replicator program
- ▶ ?ICMMOSAR? - Plan name for Asynchronous Recovery program
- ▶ ?ICMMOSDI? - Plan name for Asynchronous Delete program
- ▶ ?OWNER? - The owner of the package
- ▶ ?LSDBLOCALLOC? - Resource Manager local database location name (job ICMMRMBR)
- ▶ ?LSDBREMOTELC? - Library Server remote database location name (job ICMMRMBR)

**Note:** When installing Resource Manager in another DB2 subsystem in the same LPAR, both variables ?LSDBLOCALLOC? and ?LSDBREMOTELC? must differ for each Resource Manager.

## ICMMRMGT

This job grants access to DB2 plans.

The following variables need to be configured and must be unique for each Resource Manager:

- ▶ ?ICMMOPLN? - Plan name for Resource Manager base DB2
- ▶ ?ICMMOSAP? - Plan name for Resource Manager asynchronous replicate
- ▶ ?ICMMOSAR? - Plan name for Resource Manager asynchronous recovery
- ▶ ?ICMMOSDI? - Plan name for Resource Manager asynchronous delete

**Note:** When installing Resource Manager in *another* DB2 subsystem in the same LPAR, the values may be the same for another Resource Manager database installed in a different DB2 subsystem.

## HTTP server

You should make sure the `httpd.env`, `httpd.conf`, and the procedure name (sample default `ICMMRMWB`) are different. As `httpd.env` and `httpd.conf` reside in z/OS UNIX System Services (USS), we can put the port number as suffix to distinguish the different files (for example, `httpd.env.1080` and `httpd.conf.1080`). The `//SYSIN DD` statement in the HTTP procedure (sample `ICMMRMWB`) should point to different data sets or member names (the default member name is `ICMMJLWS`).

**Note:** When installing Resource Manager in *another* DB2 subsystem in the same LPAR, the same procedure for HTTP Server also applies for Resource Manager database in different DB2 subsystems but in the same LPAR.

**Note:** The variables `?ICM?`, `?DSN?`, and `?DSNRUN?` are related to libraries where the load modules for Content Manager and DB2 subsystem reside.

We do not discuss these because they may vary for each environment. For example, you may use the same load libraries for the system or you may not. It may be helpful to have distinct load libraries for each system because if you make a mistake in a load library, you may compromise all the systems that are dependent from that load library. This applies for both Library Server and Resource Manager.

## 11.4 Configure Content Manager clients

Content Manager clients include both the system administration client and Windows client. To make the Content Manager clients recognize these new Content Manager systems, we first need to catalog *again* the DB2 databases.

If you installed Content Manager in the same DB2 subsystem, you must catalog *again* the same information (host name, port number, and location name). The only thing that changes is the alias name you give to this new catalog procedure.

After running the DB2 catalog command, we need to run the server configuration utility (in your workstation machine where you installed the system administration client) to register these new Content Manager systems to the clients. See Figure 11-1.

**Server Configuration Utility**

This utility configures connections to your library servers. If you want to configure a new connection to a library server, then enter the requested information below.

If you want to view the information about other library server connections that have already been configured, then click "View Connection Information".

**View Connection Information**

Server type: DB2 Content Manager

Server name: \* DB2

Server repository type: DB2

Schema name: \* IFVTI

Host name: CTFMVS97.RALEIGH.IBM.COM

Operating system: OS/390

Port number: 3502

Remote database name: DB2

Node name: CTFMVS97.RALEIGH.IBM.COM

☐ Enable single sign on

Security options: ☐ Client authentication ☒ Server authentication

Database: User ID: ICMUSR2

Password: \*\*\*\*\*

OK Cancel Apply

Figure 11-1 Server configuration utility

Enter the fields with following information as described in Table 11-1.

Table 11-1 Field input for server configuration utility

Field	Information
Server type	It may be Content Manager or Information Integrator for Content. We must select Content Manager.

Field	Information
Server name	The name for the Content Manager. It must be the same database alias name that you used to catalogue the Library Server DB2 database.
Server repository type	We have four options: DB2, DB2CON, ORACLE, or ORACON. In the Content Manager for z/OS, the options ORACLE and ORACON do not apply. You may choose between DB2 or DB2CON. For more information about these options, refer to the manual, <i>IBM DB2 Content Manager for z/OS V8.3: System Administration Guide</i> , SC27-1335, Chapter 14, "Troubleshooting System administration", section "Troubleshooting user authentication and access control".
Schema name	The creator ID for Library Server tables.
Host name	The IP address or the host name from Library Server.
Operating system	In our scenario, it should be OS/390.
Port number	The DB2 port number.
Remote database name	The name of the remote database name. For DB2 z/OS, it may any name. We recommend that you use the same name as the field server name.
Node name	For Content Manager for z/OS, it is the same host name value.
Enable single sign-on	Check this box if you use single sign-on.
Security options	It is the authentication type for the database.
Userid	The connector user ID. If the Content Manager user does not have a system user ID in z/OS, it will use this user ID to access the system.
Password	Password for the connector user ID.

After entering the information in these fields, click **Apply** or **OK**. This updates the following files located at <installation drive>\icmroot\cmgmt\connectors:

- ▶ cmbicmsrvs.ini
- ▶ cmbicmenv.ini

cmbicmsrvs.ini contains information about the servers to connect. It includes the information you entered in the server configuration utility. Example 11-1 shows a sample cmbicmsrvs.ini file.

*Example 11-1 Sample cmbicmsrvs.ini file*

---

```
ICMSERVER=DB2I
ICMSERVERREPTYPE=DB2
ICMSchema=IFVTI
ICMSSO=FALSE
ICMDBAUTH=SERVER
ICMREMOTE=TRUE
ICMHOSTNAME=CTFMVS97.RALEIGH.IBM.COM
ICMPORT=3502
ICMREMOVEDB=DB2I
ICMNODENAME=CTFMVS97.RALEIGH.IBM.COM
ICMOSTYPE=390
```

---

► `cmbicmenv.ini`

*cmbicmenv.ini* contains the encrypted connector user ID and password. Example 11-2 shows a sample file content.

*Example 11-2 Sample cmbicmenv.ini file*

---

```
DB2I=(SUNNVVNSMjtpY21jb25uNg==)
```

---

### **Test the connection**

After installing the clients and configuring the clients, make sure you test the connection for each new system you have created.

## **11.5 XML export**

With Content Manager V8.3, it is possible to export the definitions made in one environment and import them to another environment using the system administration client.

This may be helpful if you set up two Content Manager systems, one for development and one for testing. You can export the system information from the development Content Manager system to a testing Content Manager system. The definitions are in XML format.



## Considerations when using sysplex and multiple LPARs

This chapter discusses what to consider when you set up a Content Manager system on z/OS using sysplex and multiple LPARs.

We cover the following topics:

- ▶ Parallel Sysplex overview
- ▶ Multiple LPARs accessing shared data
- ▶ HTTP servers and task control blocks
- ▶ Resource Manager load on multiple HTTP servers
- ▶ Load balancing in a sysplex environment
- ▶ TCP/IP, Virtual IP Address, and workload balancing
- ▶ Object ownership in a sysplex environment

## 12.1 Parallel Sysplex overview

The z/OS operating system takes advantage of the self-healing attributes of the hardware, and extends them by adding functions such as recovery services for all operating system code, address space isolation, and storage key protection. Functions such as Workload Manager (WLM), Resource Recovery Services (RRS), and Automatic Restart Manager (ARM) assure the availability of applications.

z/OS operating systems can be configured into a *Parallel Sysplex*®, which is the clustering technology for mainframes. The main objective of a Parallel Sysplex is continuous availability without compromising perceived client performance. This technology implements a data-sharing design that allows a database to be concurrently read and updated by application clones running on multiple z/OS images on one or more physical servers. Continuous availability avoids or minimizes unplanned outages (high availability) and reduces or eliminates planned outages.

Workload Manager (WLM) balances application workload across the systems in the Parallel Sysplex. If there is a failure or a planned outage on one system, other systems within the Parallel Sysplex take over the full workload automatically.

Workload Manager is also the key to scalability of Content Manager components running on z/OS. WLM makes effective use of a Parallel Sysplex. A Parallel Sysplex has mechanisms for leveling the load across different z/OS systems through dynamic workload distribution algorithms. WLM eliminates the need to manage each individual z/OS image in a sysplex by providing a unique global performance policy. It also helps the capacity planning effort because it ensures work is sent to the systems with available capacity.

Increased capacity and adaptability to new workloads are also benefits of WLM. WLM provides a way to increase the number of systems and the type of workloads in its installation without greatly increasing the skill level or number of staff necessary to manage the environment.

## 12.2 Multiple LPARs accessing shared data

If you have multiple instances of the same Library Server, each in its own LPAR, that need to access the common Library Server database (using logically one Library Server implementation), then you also must have an instance of the Resource Manager in each LPAR in which an instance of the Library Server is located. Each Resource Manager instance accesses a common Resource Manager database. Each of these LPARs may be on different zSeries® systems as well.

Figure 12-1 illustrates this implementation.

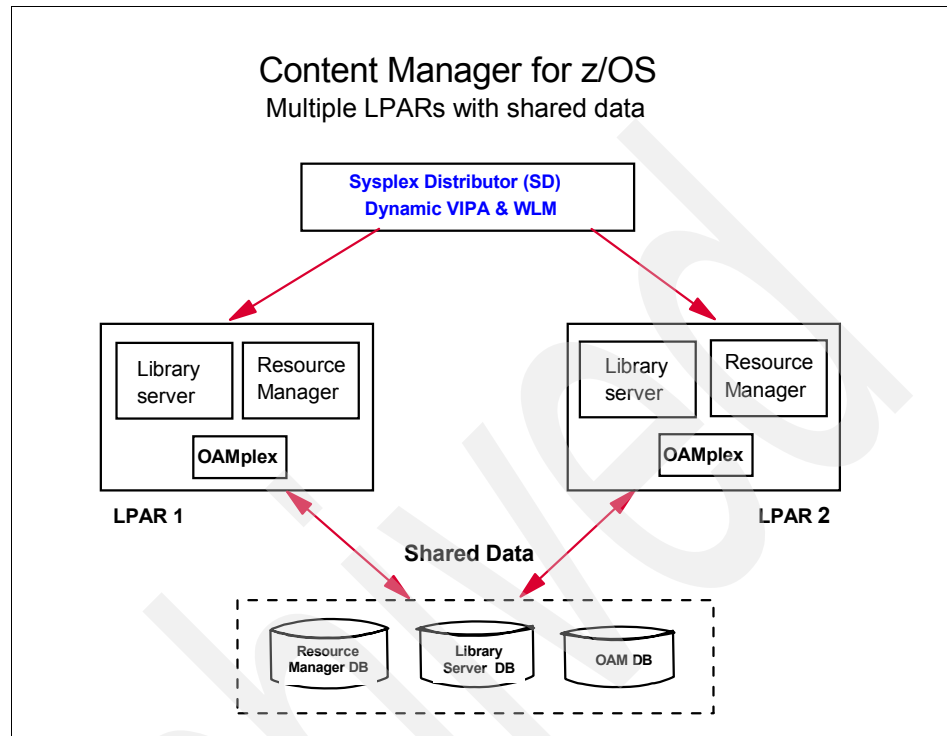


Figure 12-1 Multiple LPARs accessing shared database

Details on setting up this environment can be found in the following manuals:

- ▶ *DB2 V7 for OS/390 & z/OS - Data Sharing: Planning and Administration*, SC26-9935
- ▶ *DB2 UDB for z/OS - Data Sharing: Planning and Administration*, SC18-7417

## 12.3 HTTP servers and task control blocks

It is important to note that the Resource Manager is a CGI application running under HTTP and as such, the scalability, capacity, and availability features of HTTP are utilized.

When performing the performance benchmarks for Content Manager for z/OS, the benchmark team specified 32 TCBs (Task Control Blocks) for each HTTP server instance. While running the benchmarks, the number of TCBs per HTTP server were kept constant. When all the TCBs for a given HTTP Server are busy,

the Workload Manager will automatically start a new server instance in its own OMVS address space.

Each TCB handles a request from an active Content Manager user. When running the tests, the load indicated that the number of HTTP server instances topped out about 5 or 6 for about 1600 concurrent users in the benchmarks. It is, of course, important to note that the number of HTTP server instances vary for a given workload based on the type of actions performed by the users.

Based on the observations from the benchmark tests, you can extrapolate that using a workload similar to the samples used in the benchmark that an HTTP server instance can support approximately 300 users equivalent to one TCB per 10 concurrent users.

These suggested numbers should only be used as initial planning figures. The actual numbers should be based on local measurements using the specific workload in a given environment.

## 12.4 Resource Manager load on multiple HTTP servers

Routing of Resource Manager requests to multiple instances of the HTTP servers is handled by the Workload Manager. The Resource Manager for z/OS is running as a CGI program under the HTTP server. The HTTP server is listening on a single TCP/IP port. URLs for incoming requests are no different in a Parallel Sysplex than in single system or single sysplex environments. That is, the URLs include the host name or IP address where the HTTP Server resides and the port number where the server listens.

The HTTP server itself is normally running as a started procedure, and it can be defined to WLM. When started, the Content Manager administrator can specify a WLM-defined application environment, and by doing so, associates the HTTP server with a WLM-defined service class. A service class tells WLM what the performance goals are for the work done by this server. For example, x% of requests should be completed within y seconds or milliseconds. WLM then prioritizes the server's workload according to those goals as it searches the Parallel Sysplex for resources.

The HTTP server is a multi-threaded process handling multiple incoming HTTP requests at a time. As these HTTP requests come in, WLM may create several address spaces within the Parallel Sysplex to complete them in the most efficient way possible, starting and stopping address spaces as necessary.

For example, WLM may start an address space with a maximum number of active threads set to 50. If an HTTP request comes in, and all 50 worker threads

are busy, WLM starts a new address space to handle it, and keeps this second address space around as long as necessary. Once the new address space is no longer needed (activity drops below 50 simultaneous requests for some period of time), WLM will automatically stop it.

Externally, though, this Resource Manager is still defined to Content Manager (and to z/OS) as a single instance of the HTTP server listening at a single port.

## 12.5 Load balancing in a sysplex environment

In a Content Manager system, there are basically two types of requests sent from the clients to the servers:

- ▶ Store and search of metadata and results list, using DRDA via DB2 Client Connect to access the Library Server
- ▶ Store and retrieval of objects to the Resource Manager HTTP server using URL requests

Library Server is implemented with stored DB2 procedures. Sysplex Distributor, Dynamic Virtual IP Address (VIPA), and Workload Manager (WLM) work together to achieve load balancing for the Library Server requests as follows:

1. DRDA protocol (Level 3 clients, DB2 7.2), from DB2 clients, initially access the Sysplex Distributor.
2. Sysplex Distributor routes the request to a DB2 z/OS instance.
3. The initial DB2 instance returns a list of IP addresses of other DB2 z/OS instances and relative priorities, from WLM, that the DB2 client can use for routing subsequent DRDA connection requests.

As described above, the DRDA connection's balancing is mainly executed by the DRDA and WLM function, not the Sysplex Distributor.

Resource Manager on z/OS is implemented as a CGI program under a HTTP server on z/OS. Sysplex Distributor, Dynamic VIPA, and WLM work together to achieve load balancing for the Resource Manager requests as follows:

1. HTTP protocol from the Web browser accesses the Sysplex Distributor for all connections.
2. Sysplex Distributor utilizes WLM.
3. WLM prioritizes the workload according to defined performance goals for the workload.
4. HTTP server is associated with a defined WLM service class that defines the performance goals for the work to be done by this server.

5. As requests come in, WLM may create additional HTTP address spaces to complete requests efficiently.

## 12.6 TCP/IP, Virtual IP Address, and workload balancing

Prior to APAR PQ46659, there used to be only one way to perform TCP/IP workload balancing by setting up a WLM DNS on z/OS. This approach was called *DNS routing*.

A new approach eliminates the WLM DNS requirement and uses new functions in z/OS called *Sysplex Distributor* and *Dynamic VIPA* to perform similar functionality. This significantly changes how the DDF TCP/IP listener works and how TCP/IP is configured for DB2 in a data-sharing environment.

Information on how to take advantage of Dynamic Virtual IP Address (DVIPA) and workload balancing with DB2 data-sharing in a sysplex environment can be found in the following references:

- *OS/390 and z/OS TCP/IP in the Parallel Sysplex Environment*, GM13-0026.

This document can be downloaded via the following Web site:

<http://www.ibm.com/servers/eserver/zseries/library/techpapers/pdf/gm130026.pdf>

Some of the topics discussed in this document include:

- Virtual IP Addresses
- DNS and WLM
- Sysplex Sockets

- *IBM Communications Server - IP Configuration Guide*, SC31-8725

The document can be found at:

<http://publibz.boulder.ibm.com/epubs/pdf/flaf7032.pdf>

## 12.7 Object ownership in a sysplex environment

When an object is stored in Content Manager, information about its metadata (key fields) as well as internal system-related information (such as size, creation timestamp, Resource Manager code, and SMS collection code) is stored with its corresponding item in the Content Manager Library Server.

To retrieve the object later, the client application first retrieves the item location information from the Library Server tables. Based on this information, the client

application can build the URL that is sent to the Resource Manager to retrieve the item content.

In a Parallel Sysplex scenario, there is still only one Resource Manager code associated with all objects stored within the Parallel Sysplex. Objects are not tied to a specific Workload Manager (WLM)-created instance of the HTTP server running the Resource Manager code.

When WLM creates several instances of the HTTP server to handle increased workload caused by incoming requests, those instances may be running in multiple systems within the Parallel Sysplex. There may also be multiple DB2 subsystems associated with these systems. DB2 employs the coupling technology of the Parallel Sysplex to allow data-sharing among various z/OS systems and DB2 subsystems. As long as the Resource Managers' DB2 plan is bound with the sysplex identifier tying these multiple DB2 subsystems together, it does not matter which system in the sysplex stores the object, or later tries to retrieve it. The object is accessible from all systems. DB2, in combination with the Parallel Sysplex, coupling hardware, and software, ensures the integrity of this data.

## 12.8 Additional references

In addition to the information in the Content Manager planning and installation publications, you may find additional information from the following references. Each reference discusses various additional elements of the Content Manager environment:

- ▶ *Distributed functions of DB2 for z/OS*, SG24-6952  
This IBM Redbook contains information on Workload Management for DB2 and Connection Pooling.
- ▶ Technical information on Workload Manager configuration is at the following Web site:

<http://www.ibm.com/servers/eserver/zseries/zos/wlm/>





## Access to DB2 plans and packages for Content Manager

This chapter documents how Library Server and Resource Manager are bound to DB2, and how you can change the suggested GRANT PUBLIC access default used by the installation process.

The chapter is split into separate parts for Library Server and Resource Manager; and for each part, we cover the following topics:

- ▶ Packages and plans.
- ▶ How to determine which users need access.
- ▶ GRANT access to selected users on CM components.

At the end of the chapter, we look at error situations you might encounter, and where to find information to help you analyze the situations.

## 13.1 Introduction

In order to adapt the standard Content Manager installation to your requirements with regard to DB2 security, it is necessary for you to understand how the DB2 components of Library Server and Resource Manager are accessed.

The purpose of this chapter is to give you an understanding of Library Server and Resource Manager implementation seen from a DB2 point of view. This enables you to change the way permissions are granted in the installation jobs to suit your local standards and requirements.

### 13.1.1 Why you may want to alter the defaults

When you run the installation jobs for Library Server and Resource Manager respectively, you will see the use of the DB2 permission GRANT xxx TO PUBLIC on tables, views, and program components.

You may need to change this due to requirements defined by your system administration or security department.

The following sections will enable you to make decisions on how to change the default security suggested by the installation jobs.

### 13.1.2 Names used for DB2 plans

The names for the DB2 Plans and packages in the following sections are the default names suggested by the installation job where provided. During tailoring of Library Server and Resource Manager, you will have changed these to reflect your own naming standards and requirements.

## 13.2 Library Server packages and plans

When installing the Library Server components, you use the installation job ICMMLSBD and input to this job in member ICMMBIND to bind the DB2 packages and plans used by Library Server.

In addition, you use installation job ICMXBIND to bind the package collection and plan ICMXDFUR.

### 13.2.1 Library Server DB2 packages

Library Server is implemented based on a large number of DB2 stored procedures (SP) and functions. All of these SPs and functions are defined as

EXTERNAL. EXTERNAL means that they are implemented through external programs that will be executed under the control of the selected Workload Manager (WLM).

The ICMMLSD job binds the DBRMs for all of the programs used (approximately 130 programs) into a package collection, see Table 13-1. In addition, a package collection for the batch user import utility is created. You choose the names of these collections at installation time.

Table 13-1 Library Server packages

Package	DBRM members	Qualifier	Remarks
?DB2PKGCOLLID?	ICMPLSxx	LS schema	Programs called by Library Server DB2 stored procedures and functions. The package list will include all the Library Server DBRMs: see installation member ICMMBIND for the full list of DBRMs that are included.
ICMXDFUR	ICMLOGON ICMDFUSR	LS schema	Program used in batch to import/sync user definitions in CM.

Notice that the collection name ?DB2PKGCOLLID? is also used:

- ▶ When binding the Library Server plans.
- ▶ When defining the stored procedures in job ICMMLSCR. The COLLID parameter references the collection to determine which DBRM to use for the external programs executed in the WLM.

### 13.2.2 Library Server DB2 plans

In Table 13-2, the plans used by Library Server (LS) are listed. The names are selected at installation time.

Table 13-2 Library Server DB2 plans and usage

Plan	DBRM members	Packages	Qualifier	Remarks
?SAPPLAN?		?DB2PKGCOLLID?.*	LS schema	Plan for Library Server programs used by DB2 stored procedures and functions.

Plan	DBRM members	Packages	Qualifier	Remarks
?ACLPLAN	ICMPCACL		LS schema	Plan for program used to compile ACLs during Library Server installation and after running the batch user import/sync utility ICMMDFUR. Program is executed via batch job ICMMCACL.
ICMXDFUR		ICMXDFUR.*	LS schema	Plan used by batch job ICMMDFUR to import/sync user definitions in Content Manager.

As you note from the above table:

- ▶ There is one DB2 plan, ?SAPPLAN?, including all the “online activities”, in this example, the functions performed by Library Server while processing user requests.
- ▶ There are two DB2 plans for batch programs:
  - ?ACLPLAN? for batch program ICMPCACL. This program is executed during the installation process and after importing users to Content Manager via batch job ICMMDFUR.

In job ICMMCACL, the plan for this program is rebound every time the job runs because tables referenced by the program are dropped and recreated.

  - ICMXDFUR for batch program ICMXDFUR used to batch import users to Content Manager.

## 13.3 Grant access to Library Server DB2 components

In this section, we explain and document which users need access to the Library Server DB2 components as listed in Table 13-2.

In the installation job ICMMLSGT, the default permissions suggested by the installation process, the GRANT statements from the job, are shown in Example 13-1.

The various placeholders for names surrounded by question marks are the same used in 13.2, “Library Server packages and plans” on page 316 and you should replace them at installation time.

*Example 13-1 GRANT statements in job ICMMLSGT*

---

```

GRANT EXECUTE ON PLAN      ?ACLPLAN?    TO PUBLIC;
GRANT EXECUTE ON PLAN      ?SAPPLAN?    TO PUBLIC;

GRANT EXECUTE ON PACKAGE   ?DB2PKGCOLLID?.ICMPLSGU TO PUBLIC;
GRANT EXECUTE ON PACKAGE   ?DB2PKGCOLLID?.ICMPLSIG TO PUBLIC;
GRANT EXECUTE ON PACKAGE   ?DB2PKGCOLLID?.ICMPLSIP TO PUBLIC;
GRANT EXECUTE ON PACKAGE   ?DB2PKGCOLLID?.ICMPLSIR TO PUBLIC;
GRANT EXECUTE ON PACKAGE   ?DB2PKGCOLLID?.ICMPLSIT TO PUBLIC;
...etc

GRANT EXECUTE ON PROCEDURE ?CREATOR?.ICMBUILDCOMPTYPE TO PUBLIC;
GRANT EXECUTE ON PROCEDURE ?CREATOR?.ICMBUILDCOMPVIEW TO PUBLIC;
GRANT EXECUTE ON PROCEDURE ?CREATOR?.ICMCHECKINITEM TO PUBLIC;
GRANT EXECUTE ON PROCEDURE ?CREATOR?.ICMCHECKOUTITEM TO PUBLIC;
...etc

GRANT EXECUTE ON FUNCTION  ?CREATOR?.ICMENCRIPT TO PUBLIC;
GRANT EXECUTE ON FUNCTION  ?CREATOR?.ICMDECRYPT TO PUBLIC;
GRANT EXECUTE ON FUNCTION  ?CREATOR?.ICMACLPRIVEXIT TO PUBLIC;
GRANT EXECUTE ON FUNCTION  ?CREATOR?.ICMACLPRIVBITCHECK TO PUBLIC;

```

---

As you can see, you need to grant access to the plans and packages as well as the DB2 stored procedures and functions used by Library Server. The job ICMMLSGT includes a GRANT statement for each member of package ?DB2PKGCOLLID? as well as each stored procedure.

If you wish to change the use of GRANT PUBLIC to grant access to selected users and/or RACF groups, you may find it daunting from a maintenance point of view to maintain the permissions on every stored procedure and DBRM used by Library Server. In this case, you may choose to use generic names for the stored procedures and packages as illustrated in Example 13-2.

*Example 13-2 Generic GRANT statements in job ICMMLSGT*

---

```

GRANT EXECUTE ON PLAN      ?ACLPLAN?      TO PUBLIC;
GRANT EXECUTE ON PLAN      ?SAPPLAN?      TO PUBLIC;
GRANT EXECUTE ON PACKAGE   ?DB2PKGCOLLID?.* TO PUBLIC;
GRANT EXECUTE ON PROCEDURE ?LS_SCHEMA?.*  TO PUBLIC;
GRANT EXECUTE ON FUNCTION  ?LS_SCHEMA?.*  TO PUBLIC;

```

---

In many DB2 installations, it is not customary or maybe unacceptable to grant public access to DB2 resources. In order to be more specific in your authorizations, you need to understand how the programs using the DB2 resources are run and what users are connecting to DB2 when the programs are running.

If you do not wish to use the GRANT EXECUTE TO PUBLIC approach, you must grant access to the following users:

- ▶ Library Server administrators
- ▶ The ICMCONCT user if you use this user
- ▶ All users logging on to Library Server not using the ICMCONCT user
- ▶ Any Resource Manager user ID used to run the Asynchronous utilities

In addition, the user used during the Library Server database installation and the user used running the jobs to define new item types need DBADM access to the Library Server database.

### 13.3.1 Library Server access

To understand who needs access to the Library Server DB2 plans, packages, stored procedures, and functions, you must understand how users and programs connect to the Library Server system.

When a user logs on to Library Server, two types of connections and logons are performed at this point:

1. Establish a connection to the DB2 system where the Library Server database is installed (using the DB2 CONNECT method).
2. Logon to the Content Manager Library Server using a user ID defined to Library Server.

The user ID used for these two logons may be the same user ID or two different user IDs as discussed in Table 13-3. This is important to understand when discussing what user IDs need permissions on the DB2 resources.

The users can be split into a number of categories connecting to Library Server as listed in Table 13-3.

In the table, the term “CM Connect ID” refers to the user called ICMCONCT in the installation manual. The value for user ID and password are saved in encrypted format in the `cmbicmenv.ini` file on the client workstation during client installation and tailoring.

Refer to *IBM DB2 Content Manager Enterprise Edition V8.3: Planning and Installing Your Content Management System*, GC27-1332, (note this is the multiplatforms manual, not the z/OS manual), for details on how to create and manage the `cmbicmenv.ini` file.

Table 13-3 User types accessing the Library Server DB2 resources

User type	Access
Administration user during install	While installing Library Server, you run a utility from your workstation (cmcfgls) where you specify a user ID and password. This user must be defined as a Content Manager administrator. Note: This is the initial user you define when running the ICMMLSLD installation job - You should not depend on being able to use the system administration client to define additional administration users before installation has completed. The user will connect to DB2 and needs update access to the Library Server DB2 tables as well as the execute privilege to the Library Server plans, packages, stored procedures, and functions.
Administration user using the system administration client	These users must be defined in Content Manager as administrators. They will also connect to DB2 with the <i>same</i> user ID. These users need the execute privilege to the Library Server plans, packages, stored procedures, and functions. They need no special access to the Library Server DB2 tables. Note: When defining new item types to Content Manager, new tables are defined in the Library Server database. This is done by running batch jobs so it is the user ID used in these batch jobs that will need DBADM privilege and not the online administration user.
CM user using the Windows client without using the CM Connect ID	These users must be defined to Content Manager. They will also connect to DB2 with the <i>same</i> user ID. These users need the execute privilege to the Library Server plans, packages, stored procedures, and functions.
CM user using the CM Connect ID	These users must be defined to Content Manager. They will connect to DB2 using the CM Connect ID. For these users, only the Content Manager Connect ID needs the execute privilege on the Library Server plans, packages, stored procedures, and functions.

Using the Content Manager Connect ID can be seen as an alternative to give access to Content Manager DB2 resources to users that either have no RACF ID /DB2 CONNECT authority on the system or if you wish to limit the number of users to whom you grant access to the Content Manager DB2 resources.

### 13.3.2 CM clients logon processing to DB2

In this section, we explain the use of a tailoring parameter in the cmbicmenv.ini to control the way a user connects to the Library Server DB2 system. The text is an extract from a document you can find on IBM's support Web site:

<http://www.ibm.com/support/docview.wss?uid=swg21173808>

In summary, it explains how the parameter ICMSEVERTYPE can control the connect sequence to the DB2 system of Library Server is performed.

By default, Content Manager logon processing's first attempt to authenticate the user ID is through DB2 Universal Database™. If it is successful, it does not check the password stored in the Library Server ICMSTUSERS table.

If DB2 authentication fails, Content Manager logon processing would then use the encrypted user ID and password pair stored in *cmbicmenv.ini* (Example, ICMCONCT) to authenticate with DB2 database. Assuming the ICMCONCT user ID and password authenticates successfully to DB2 database, only then would Content Manager logon processing proceed to check the given user ID and password (passed in by the user at logon) against the corresponding user ID and password in the Library Server ICMSTUSERS table.

Note that you can modify this behavior by changing ICMSEVERTYPE from DB2 to DB2CON in your *cmbicmsrvs.ini*. ICMSEVERTYPE=DB2CON effectively reverses the order of the user IDs used for authentication during Content Manager logon processing.

With DB2CON, the user ID and password pair *in cmbicmenv.ini* (the "ICMCONCT" user ID) is checked first for authentication to DB2 database and if it authenticates successfully, then the given user ID and password (passed in from the user logon) is checked against the corresponding user ID and password value stored in the Library Server ICMSTUSERS table and no attempt is made to authenticate with DB2 database using that given user ID and password (passed in from the user logon). If the ICMCONCT authentication fails, the given user ID and password (passed in from the user logon) is used during Content Manager logon processing to attempt to authenticate with DB2 database.

The above discussion assumes a user ID for which "Use system password" is not checked when it is defined in the system administration client's Define Users window. If the "Use system password" check box was checked for the user ID, authentication would always be only through DB2 database and the user ID and password pair stored in *cmbicmenv.ini* would not be used.

### 13.3.3 Batch jobs ICMMACCL and ICMMDFUR

When running the two batch jobs, ICMMACCL and ICMMDFUR, you, at run time, specify under whose authority the job executes. Access to the DB2 plans, ?ACLPLAN? and ICMXDFUR, (see Table 13-2 on page 317) can be limited to the users executing these batch jobs.



### 13.3.4 Options used when binding Library Server programs

The Library Server programs use a combination of static and dynamic SQL to access the Library Server tables. This has a number of implications with regards to options used when binding the Library Server DB2 packages.

In Example 13-3, we show an extract from installation member ICMMBIND.

*Example 13-3 Extract from installation member ICMMBIND*

---

BIND PACKAGE (?DB2PKGCOLLID?) -
MEMBER(ICMPLSX0) -
QUALIFIER(?CREATOR?) -
OWNER(?OWNER?) -
ISOLATION(CS) -
CURRENTDATA(NO) -
RELEASE(COMMIT) -
ACTION(REPLACE) -
DYNAMICRULES(BIND)
BIND PLAN (?SAPPLAN?) -
PKLIST(?DB2PKGCOLLID?.* ) -
QUALIFIER(?CREATOR?) -
OWNER(?OWNER?) -
ISOLATION(CS) -
RELEASE(COMMIT) -
ACTION(REPLACE) -
DYNAMICRULES(BIND)

---

Notice the parameters used in the bind processing described in Table 13-4.

*Table 13-4 Options used in bind process*

BIND option	Remarks
DYNAMICRULES(BIND)	DB2 uses the authorization ID of the plan or package for authorization checking of dynamic SQL statements. Unqualified table, view, index, and alias names in dynamic SQL statements are implicitly qualified with the value of the bind option QUALIFIER; if you do not specify QUALIFIER, DB2 uses the authorization ID of the plan or package owner as the implicit qualifier.
QUALIFIER(?CREATOR?)	Use the schema name for the Library Server tables for unqualified table names.
OWNER(?OWNER?)	Determines the authorization ID of the owner of the object. The owner must have the privileges required to execute the SQL statements contained in the object.

The combined result of the above parameters is that a user connecting to the Library Server database and executing one of the Library Server stored procedures needs no personal access rights to the Library Server tables.

By granting the EXECUTE permission on the plans, packages, and stored procedures, the user is authorized to access the Library Server tables through the programs (both for the static as well as the dynamic SQL used) and perform the required accesses and updates performed by the programs.

### 13.3.5 Access to the DB2 views on Library Server tables

When installing Library Server and later when creating new item types, additional tables and views are created in the Library Server database. On these DB2 views, the standard jobs issue a GRANT SELECT PUBLIC.

Based on the information obtained from the developers, the DB2 views are not used by the Content Manager base product.

They are provided to enable your own programs to access the item type tables using a more descriptive view name. The views are based on a JOIN operation with the Content Manager security control tables. This way, a user can only see information on items the user is authorized to see.

If you have concerns giving PUBLIC access to these views, you can choose to:

- ▶ Skip the GRANT PUBLIC permission on the views if you do not plan to develop your own programs accessing these views.
- ▶ If you plan to use the views, you can limit the access to those users who need to read information from the view instead of using PUBLIC.

The rest of this section gives additional background on DB2 views to enable you to evaluate DB2 security on these views.

Example 13-4 contains an extract from the Library Server job CVW01000.

*Example 13-4 Extract from Library Server job CVW01000*

---

```
1. CREATE VIEW ICM4BLS.NOINDEX001
2. (COMPCLUSTERID, COMPONENTID, ITEMID, VERSIONID, ACLCODE, SEMANTICTYPE,
3.  EXPIRATIONDATE, COMPKEY, CREATETS, CREATEUSERID, LASTCHANGEDTS,
4.  LASTCHANGEDUSERID,
5.  SOURCE, USER_ID, TIMESTAMP)
6. AS SELECT
7.  COMPCLUSTERID, COMPONENTID, ITEMID, VERSIONID, ACLCODE, SEMANTICTYPE,
8.  EXPIRATIONDATE, COMPKEY, CREATETS, CREATEUSERID, LASTCHANGEDTS,
9.  LASTCHANGEDUSERID,
10. ATTR0000001000, ATTR0000001001, ATTR0000001002
```

```

11. FROM ICM4BLS.ICMUT01000001
12. WHERE (EXISTS
13. (SELECT 1 FROM
14.   ICM4BLS.ICMSTCOMPILEDACL AS ICMC,
15.   ICM4BLS.ICMSTUSERS      AS U
16.   WHERE (ICMC.ACL=3 OR ICMC.ACL=-1)
17.   AND U.USERID = USER
18.   AND ICMC.UNUM = U.UNUM
19.   AND ICMC.RPRIV='1' ) );

20. GRANT SELECT ON TABLE ICM4BLS.NOINDEX001 TO PUBLIC;

```

---

In Example 13-4, the view definition for the NOINDEX item type is shown. The name of the DB2 views is based on the item type name in order to provide a more user friendly name to programmers:

- ▶ The DB2 table name is ICMUT01000001.
- ▶ The DB2 view name is ICMUTNOINDEX001.

With reference to the line number, notice the following points:

- ▶ In lines 13 to 19, the view is joined with information in the Content Manager user and ACL (Access Control List) tables based on the value on the USER value. This is the CURRENT SQLID of the currently connected user. The view is therefore limited to only include rows the user is authorized to see based on the Content Manager security schemes.
- ▶ Lines 10 and 5 show how the internally used column names for key fields (metadata) are mapped to the more easy to use display names.
- ▶ In line 20, the default access PUBLIC is being granted on the view.

To summarize, granting PUBLIC access to the DB2 views defined on the Content Manager table ICMUTnnnnn001 does not pose a security risk, but you may choose to skip this if you do not plan to use the DB2 views.

## 13.4 Resource Manager packages and plans

In order to bind the DB2 packages and plans for Resource Manager, there are two separate jobs in the installation library. The job you use depends on whether your Resource Manager is running on the same DB2 system as the Library Server or on two separate DB2 systems:

- ▶ ICMMRMBD - Use this job if Resource Manager and Library Server are on same DB2 system.
- ▶ ICMMRMBR - Use this job if Resource Manager and Library Server are on two different DB2 systems.

Both jobs have two steps:

1. Bind a number of packages depending on whether the Library Server is local or remote to Resource Manager.
2. Bind the plans for Resource Manager including the packages created in the first step as well as various OAM-related packages. This step is similar for both jobs.

While executing the job, Resource Manager has to access the Library Server with which it is associated. It is in this context that it is important to note that a Resource Manager can only be associated with *one* Library Server. This will become evident when reviewing the way the packages and plans are defined and bound to DB2.

### 13.4.1 RM and LS on the same DB2 system

In this section, we describe how job ICMRMBD is used to bind packages and plans used when Resource Manager and Library Server are installed on the same DB2 system.

#### Resource Manager DB2 packages for local Library Server

When using installation job ICMRMBD, the first step in the job binds the packages listed in Table 13-5.

Table 13-5 Resource Manager packages bound when RM and LS are on the same DB2 system

Package	DBRM members	Qualifier	Remarks
?lsdbloc?.ICMMGRPC	ICMMOSDR	LS schema	Encryption key handler for Resource Manager
?lsdbloc?.ICMMGRPP	ICMMOSP1	LS schema	Subroutines of the replicator module that access Library Server database
?lsdbloc?.ICMMGRPR	ICMMOSR1	LS schema	Subroutines of the asynchronous recovery module that access Library Server database
?lsdbloc?.ICMMGRPD	ICMMOSD1	LS schema	Subroutines of the asynchronous delete process that access Library Server database

The first part of the package name ?lsdbloc? refers to your local DB2 system and it must be defined in the SYSIBM.LOCATIONS table. You may skip this prefix since the bind process will default to the local system if you do not specify a location name.

The packages will subsequently be included in the plans you bind in the second step of the installation job ICMMRMBD.

From the “Qualifier” column, the packages are referencing the schema name used when defining the Library Server tables because these programs are accessing the Library Server tables.

## Resource Manager DB2 plans

Table 13-6 lists the plans used by Resource Manager.

*Table 13-6 Resource Manager DB2 plans when RM and LS are on same DB2 system*

Plan	DBRM members	Packages	Qualifier	Remarks
ICMMOPLN	ICMMOSTT ICMMOSET CBRHTBSV	ICMMGRPC.* *.CBRIDBS	RM schema	Plan for CGI program running in the Resource Manager HTTP server. The program name is specified in the HTTP server's configuration file and the plan name is specified via a SYSIN data set in the HTTP server procedure.
ICMMOSAP	ICMMOSAP CBRHTBSV	ICMMGRPP.* *.CBRIDBS	RM schema	Plan for asynchronous replicator program. This program is run using job ICMMRMAP. In the job, you specify program and plan name as well as the user ID used to connect to DB2.
ICMMOSAR	ICMMOSAR CBRHTBSV	ICMMGRPR.* *.CBRIDBS	RM schema	Plan for asynchronous recovery program. This program is run using job ICMMRMAR. In the job, you specify the program and plan name as well as the user ID used to connect to DB2.
ICMMOSDI	ICMMOSDI CBRHTBSV	ICMMGRPD.* *.CBRIDBS	RM schema	Plan for asynchronous delete program. This program is run using job ICMMRMDI. In the job, you specify the program and plan name as well as the user ID used to connect to DB2.

Note in the “Package” column, the respective plans include the following packages:

- ▶ The generic ICMMGRPx.\* packages bound in the first step of the job.
- ▶ The generic \*.CBRIDBS packages in order to access the OAM storage groups. The installation jobs suggest referencing specific OAM packages using a number of GROUPnn.CBRIDBS references. You may also choose to use a

generic reference \*.CBRIDBS as shown. If you use the generic method, you do not have to rebind when adding new storage groups.

## 13.4.2 RM and LS on separate DB2 systems

In this section, we describe how job ICMRMBR is used to bind the packages and plans used when Resource Manager and Library Server are installed on two different DB2 systems.

### Resource Manager DB2 packages for remote Library Server

When using installation job ICMRMBR, the first step in the job binds the packages listed in Table 13-7.

Table 13-7 Resource Manager packages bound when RM and LS are on different DB2 systems

Package	DBRM members	Qualifier	Remarks
?lsdbloc?.?lsschema?	ICMMOSDR	RM schema	Encryption key handler for Resource Manager
?lsdbremoteloc?.?lsschema?	ICMMOSDR	LS schema	
?lsdbloc?.?lsschema?	ICMMOSP1	RM schema	Subroutines of the replicator module that access Library Server database
?lsdbremoteloc?.?lsschema?	ICMMOSP1	LS schema	
?lsdbloc?.?lsschema?	ICMMOSR1	RM schema	Subroutines of the asynchronous recovery module that access Library Server database
?lsdbremoteloc?.?lsschema?	ICMMOSR1	LS schema	
?lsdbloc?.?lsschema?	ICMMOSD1	RM schema	Subroutines of the asynchronous delete process that access Library Server database
?lsdbremoteloc?.?lsschema?	ICMMOSD1	LS schema	

Note the differences between this scenario and the one shown earlier in Table 13-5 on page 326:

- When Resource Manager is on the same DB2 system as Library Server, you only bind the Resource Manager packages on the local system.
- When Library Server is remote, you bind the packages on both the local and remote DB2 systems.

The packages will subsequently be included in the plans you bind in the second step of the installation job ICMRMBR.

From the “Qualifier” column, note:

- The remote packages are referencing the schema name used when defining the Library Server tables in order to access the Library Server tables.

- The local packages are referencing the schema name for the Resource Manager tables in order to access the local Resource Manager tables.

## Details on how the RM packages are bound

Example 13-5 is an extract from the first step in the ICMMRMBR job.

*Example 13-5 Extract from job ICMMRMBR to bind RM packages with a remote LS*

---

```
1.  BIND PACKAGE(?LSDBREMOTELOC?.?LSSCHEMA?) -
2.      MEMBER(ICMMOSP1) -
3.      QUALIFIER(?LSSCHEMA?) -
4.      OWNER(?OWNER?) -
5.      ISOLATION(CS) -
6.      RELEASE(COMMIT) -
7.      SQLERROR(CONTINUE) -
8.      VALIDATE(RUN) -
9.      ACTION(REPLACE)

10. BIND PACKAGE(?LSDBLOCALLOC?.?LSSCHEMA?) -
11.     MEMBER(ICMMOSP1) -
12.     QUALIFIER(?CREATOR?) -
13.     OWNER(?OWNER?) -
14.     ISOLATION(CS) -
15.     RELEASE(COMMIT) -
16.     SQLERROR(CONTINUE) -
17.     VALIDATE(RUN) -
18.     ACTION(REPLACE)
```

---

Example 13-5 shows the two BIND statements for one of the Resource Manager DBRMs. The job ICMMRMBR contains such a pair of BIND statements for each of the DBRMs used by Resource Manager in order to bind it both on the local as well as the remote DB2 system.

When you look at the BIND PACKAGE statements, notice the following points:

- Lines 1 and 10: When binding Resource Manager to be used with a remote Library Server, you have to bind the package both on the local as well as the remote system. The package name is built based on two parts separated by a full stop character:
  - The first part is the location name for the DBMS system where the package is bound and where the description of the package resides. In line 1, we bind to the remote DB2 system and in line 10 to the local DB2 system, where Resource Manager is running.

The location name must be defined in catalog table, SYSIBM.LOCATIONS. If that table does not exist or if the DBMS is not in it, you receive an error message.

- The second part is the package name. Do not get confused by the package name ?LSSCHEMA? suggested by the installation job.

The package name has no dependency nor connection to the actual schema name used by Library Server. It is just a convenient way for the installation job to suggest a name for the package. You may change this package name independently of the schema name used in the QUALIFIER statement in lines 3 and 12. If you do decide to change it, make sure you change it in *all* the PACKAGE statements in the job as well as the reference in the BIND PLAN statements in the second step of the job.

- ▶ In lines 2 and 11, the member ICMOSP1 is used in both package BINDs. In other words, you are binding the same DBRM on both the local and the remote system.
- ▶ In lines 3 and 12, the QUALIFIER statement defines the schema name for the tables to access by the program. Two different schema names are used:
  - In line 3, the package bound on the remote Library Server system uses the Library Server schema name.
  - In line 12, the package bound on the local Resource Manager system refers to the Resource Manager schema name.
- ▶ In lines 4 and 13, the owner of the package is specified. The package owner must have authorization to execute all the statements embedded in the package for BIND PACKAGE to build a package without producing error messages.

## Resource Manager DB2 plans

Table 13-8 shows the plans used by Resource Manager.

Table 13-8 Resource Manager DB2 plans and usage

Plan	DBRM members	Packages	Qualifier	Remarks
ICMMOPLN	ICMMOSTT ICMMOSET CBRHTBSV	*.?LSSCHEMA?.* *.CBRIDBS	RM schema	Plan for CGI program running in the Resource Manager HTTP server. The program name is specified in the HTTP server's configuration file and the plan name is specified via a SYSIN data set in the HTTP server procedure.
ICMMOSAP	ICMMOSAP CBRHTBSV	*.?LSSCHEMA?.* *.CBRIDBS	RM schema	Plan for asynchronous replicator program. This program is run using job ICMRMAP. In the job, you specify program and plan name as well as the user ID used to connect to DB2.



Plan	DBRM members	Packages	Qualifier	Remarks
ICMMOSAR	ICMMOSAR CBRHTBSV	*.?LSSCHEMA?.* *.CBRIDBS	RM schema	Plan for asynchronous recovery program. This program is run using job ICMMRMAR. In the job, you specify the program and plan name as well as the user ID used to connect to DB2.
ICMMOSDI	ICMMOSDI CBRHTBSV	*.?LSSCHEMA?.* *.CBRIDBS	RM schema	Plan for Asynchronous delete program. This program is run using job ICMMRMDI. In the job, you specify the program and plan name as well as the user ID used to connect to DB2.

Note, in the “Package” column, the respective plans include the following packages:

- ▶ The generic `*.?LSSCHEMA?.*` packages bound in the first step.
- ▶ The generic `*.CBRIDBS` packages in order to access the OAM storage groups.

The installation jobs suggest referencing specific OAM packages using a number of `GROUPnn.CBRIDBS` references, but you may also choose to use a generic reference `*.CBRIDBS` as shown. If you use the generic method, you do not have to rebind when adding new storage groups.

### Details on how the RM plans are bound

The second step of the ICMMRMBR job binds the plans for Resource Manager. Example 13-6 is an extract from the job where the plan using the package shown in Example 13-5 on page 329 is bound.

*Example 13-6 Extract from job ICMMRMBR to bind a RM Plan when using remote LS*

---

```

1. BIND PLAN (?ICMMOSAP?) -
2.     MEMBER(ICMMOSAP, CBRHTBSV) -
3.     PKLIST(*.?LSSCHEMA?.*, -
4.           ?GROUP00?.CBRIDBS, -
5.           ?GROUPnn?.CBRIDBS) -
6.     QUALIFIER(?CREATOR?) -
7.     VALIDATE(BIND) -
8.     OWNER(?OWNER?) -
9.     ISOLATION(CS) -
10.    CURRENTDATA(NO) -
11.    RELEASE(COMMIT) -
12.    ACTION(REPLACE) -
13.    ENABLE(*)

```

---

In Example 13-6, when binding one of the Resource Manager plans, note:

- In line 3, the list of packages includes the generic `*.?LSSCHEMSA?.*` (note the leading and trailing asterisks). This way both the package for the local as well as the remote LS are included.

If you decide to change the package name to something other than the suggested `?LSSCHEMA?` value when creating the packages, you need to change it here, too.

## 13.5 Grant access to RM packages and plans

In the previous sections, we show a list of the packages and plans used by the Resource Manager. The question now is: Who needs access to these when Resource Manager is running?

The default job, `ICMMRMGT`, supplied in the installation library as a default, grants the `EXECUTE` right on the plans and packages to `PUBLIC`. A copy of the DB2 statements can be seen in Example 13-7.

*Example 13-7 GRANT statements in job ICMMRMGT*

---

```
GRANT EXECUTE ON PLAN ?ICMMOPLN? TO PUBLIC;  
GRANT EXECUTE ON PLAN ?ICMMOSAP? TO PUBLIC;  
GRANT EXECUTE ON PLAN ?ICMMOSAR? TO PUBLIC;  
GRANT EXECUTE ON PLAN ?ICMMOSDI? TO PUBLIC;
```

---

In many installations, it is not customary or maybe unacceptable to grant `PUBLIC` access to DB2 resources. In order to be more specific in our authorizations, we need to understand how the programs using the DB2 resources are run and which users are connecting to DB2 when the programs are running.

### 13.5.1 Resource Manager batch programs

In Table 13-9, we list the Resource Manager programs that are executed using separate batch jobs.

*Table 13-9 List of Resource Manager batch jobs*

Program name	DB2 plan	Sample job using program
ICMMOSAP	?ICMMOSAP?	ICMMRMAP
ICMMOSAR	?ICMMOSAR?	ICMMRMAR
ICMMOSDI	?ICMMOSDI?	ICMMRMDI

In these batch jobs, you specify the program name, the DB2 plan, as well as the user ID used by the program to connect to DB2. You may therefore limit the grants used in installation job, ICMRMGT, to include the users you plan to use in these batch programs.

In Example 13-8, it contains the sample parameter string passed from the job, ICMRMGT, to the program, ICMOSAP, in the EXEC statement.

*Example 13-8 Sample parameter string passed to ICMOSAP in job ICMRMGT*

---

```
//ICMOSAP EXEC PGM=ICMOSAP,DYNAMNR=20,REGION=0M,TIME=NOLIMIT,
//      PARM=('?DB2SYS?','?LSDBLOC?','?LOCAL/REMOTE?','?ICMOSAP?',
//      '?RMNAME?','?SLEEP?','?TRACE?','?TARGETRM?')
```

---

A detailed explanation of the various parameters can be found in the comments of the job.

## 13.5.2 Resource Manager CGI program running in the HTTP server

The program implementing the Resource Manager's "online" process (as opposed to the batch activities mentioned in the previous section) runs as a CGI program under control of a HTTP server task.

A HTTP server configuration parameter has an impact on how the CGI program connects to DB2. This parameter is specified in the configuration file referenced in the PARM statement of the HTTP server procedure. Example 13-9 shows an example from the sample job, ICMRMWB, where the configuration file is specified via the "-r" parameter.

*Example 13-9 Example from ICMRMWB sample procedure*

---

```
//ICMRMWB PROC ICSPARM='-p ?PORT? -r /etc/icmrmcf.conf',
//      LEPARM='ENVAR("_CEE_ENVFILE=/etc/httpd.envvars"),ALL31(ON)'
//WEBSRV EXEC PGM=IMWHTTPD,REGION=0K,TIME=NOLIMIT,
//      PARM=('&LEPARM/&ICSPARM')
```

---

Among many values you specify in the configuration file, the "UserId" has special interest when it comes to security. Table 13-10 shows two of the possible values for the "UserId" parameter and how they influence DB2 access.

*Table 13-10 Setting the UserId value in the HTTP server configuration file*

Parameter	Description
UserId %%SERVER%%	If you specify "server", the CGI program will use the user ID that the HTTP server started task is running under to connect to DB2.

Parameter	Description
UserId %%CLIENT%%	If you specify “client”, the user ID passed by the caller will be used to connect to DB2.

**Note:** The values shown in the parameter column of Table 13-10 are shown as they are specified in the configuration file.

The values include the two leading and trailing percentage signs.

Some additional comments on using the “UserId %%CLIENT%%” setting:

- ▶ Library Server passes the user ID specified for the Resource Manager connection user ID when communicating with Resource Manager. This is done either when Library Server communicates directly via HTTP to Resource Manager and when a user requests a document from Resource Manager using a URL with a link to a document; the URL includes the user ID and password in an encrypted field of the URL string.
- ▶ You specify the initial setting for the Resource Manager user ID when running the Library Server installation job ICMMLSLD. The value is loaded into column RMUserID of table ICMSTResourceMgr.
- ▶ You can see this user ID using the system administration client when showing the “properties” for a given Resource Manager definition. You can also specify and update both the user ID as well as the password.

**Important:** When you run the Library Server installation job, ICMMLSLD, a dummy password is set for the Resource Manager user in table, ICMSTResourceManager. You must set the correct password for this user using the administration client before attempting to store and access data in the Resource Manager; otherwise, if you use the “%%CLIENT%%” setting, your access to Resource Manager will fail.

Based on the above information, if you wish to change the suggested GRANT privilege of PUBLIC (default) on the CGI program, you must grant access to either the user running the HTTP server and/or the Resource Manager connect user defined in the Library Server definition for the Resource Manager.

For a detailed description of the HTTP server configuration file, refer to the manual *z/OS HTTP Server Planning, Installing, and Using*, SC34-4826.

### 13.5.3 Summary for RM GRANT specification

We can summarize the DB2 GRANT requirements as shown in Table 13-11.

Table 13-11 Summary of Resource Manager DB2 GRANT requirements

DB2 plan	Grant access to:
?ICMMOPLN?	User executing the HTTP server procedure -- or -- User defined as Resource Manager access user in the Library Server. This depends on your HTTP configuration.
?ICMMOSAP?	Users running the ICMRMAP job.
?ICMMOSAR?	Users running the ICMRMAR job.
?ICMMOSDI?	Users running the ICMRMDI job.

## 13.6 LS problems related to access security

In this section, we document various problems you may encounter if you have problems related to how you have set up your security on your Library Server system.

Some of these are problems we encountered when we installed our test system. Others we provoked by changing various settings. The descriptions of these situations should help you to debug your own situations in case of problems.

We use the user ID ICMCONCT for the name of the connect user. You may have chosen another name for this user ID during the installation.

### 13.6.1 Invalid values for connect user ICMCONCT

In this section, we list some of the error messages you encounter if there is a problem using the Content Manager connect user ID from the cmbicmenv.ini file.

The messages in the examples are copied from the error message window of the Windows client.

If the ICMCONCT user ID has been revoked, you will see the error message shown in Example 13-10 displayed by the Windows client. You will also see the accompanying message from RACF in the system log.

*Example 13-10 Windows client error if ICMCONCT user has been revoked*

---

The logon was not successful. See the Details section below for more information.

DGL0394A: Error in :Error - SQL\_ERROR

```
[IBM][CLI Driver] SQL30082N Attempt to establish connection failed with
security reason "19" ("USERID DISABLED or RESTRICTED").  SQLSTATE=08001
(STATE) : 08001[SERVER=ICM4BLSD:USERID=ICMCONCT] (STATE) : 08001
```

---

If you have let the password for the ICMCONCT user expired, users will see the error message shown in Example 13-11.

*Example 13-11 Windows client error if ICMCONCT user password is expired*

---

The logon was not successful. See the Details section below for more information.

```
DGL0394A: Error in :Error - SQL_ERROR
[IBM][CLI Driver] SQL30082N Attempt to establish connection failed with
security reason "1" ("PASSWORD EXPIRED").  SQLSTATE=08001
(STATE) : 08001[SERVER=ICM4BLSD:USERID=ICMCONCT] (STATE) : 08001
```

---

If you supply an invalid password in the `cmbicmenv.ini` file, users will see the error message shown in Example 13-12. You will also see the accompanying message from RACF in the system log.

This situation can arise if you change the RACF password of the ICMCONCT user without updating the users in `cmbicmenv.ini` files. In this case, the users will try to connect to DB2 using the ICMCONCT user's old password saved in this tailoring file.

**Important:** If you do not update the `cmbicmenv.ini` files used by all the users, one user may cause the ICMCONCT user to be RACF-revoked through too many invalid logon attempts.

Depending on your RACF configuration, you may not be able to resume the ICMCONCT user ID again with the same password. In this case, you have to distribute a fresh `cmbicmenv.ini` file to *all* users using the Content Manager connect user.

*Example 13-12 Windows client error if ICMCONCT password is invalid*

---

The logon was not successful. See the Details section below for more information

```
DGL0394A: Error in :Error - SQL_ERROR
[IBM][CLI Driver] SQL30082N Attempt to establish connection failed with
security reason "24" ("USERNAME AND/OR PASSWORD INVALID").  SQLSTATE=08001
(STATE) : 08001[SERVER=ICM4BLSD:USERID=ICMCONCT] (STATE) : 08001
```

---

## 13.7 RM problems related to access security

In this section, we document various problems you may encounter if you have problems related to how you have set up your security on your Resource Manager system. Some of these are problems we encountered during our test installation and others we provoked by changing various settings. The descriptions of these situations should help you to debug your own situations in case of problems.

### 13.7.1 Authority to write to DSNTRACE

If you need to activate the DB2 trace by adding the DSNTRACE DD statement to the HTTP server startup procedure as described in the Chapter “Enabling the Resource Manager DB2 trace facility DSNTRACE” in the IBM DB2 Content Manager installation manual, you may encounter a problem if you use the “UserId %%CLIENT%%” setting in your HTTP configuration file.

When using this setting, the entries are written to the DSNTRACE file under the authority of the Resource Manager user ID passed by Library Server. If this user is not authorized to write to the spool system, you may see an error during HTTP server startup as shown in Example 13-13.

*Example 13-13 Error writing to DSNTRACE file due to missing authority*

---

```
16.24.33 STC05095 ---- WEDNESDAY, 16 MAR 2005 ----
16.24.33 STC05095 IEF695I START ICM830B4 WITH JOBNAME ICM830B4 IS ASSIGNED TO
USER ICMST , GROUP STCGRP
16.24.33 STC05095 $HASP373 ICM830B4 STARTED
16.24.33 STC05095 IEF403I ICM830B4 - STARTED - TIME=16.24.33
16.24.33 STC05095 IMW3534I PID: 67437012 SERVER STARTING
16.32.10 STC05095 IMW3536I SA 67437012 0.0.0.0:8088 ** READY
16.32.58 STC05095 ICH408I USER(ICMRM ) GROUP(ICM ) NAME(ICM RM SERV)
807
807 CPHMVSG.ICMST.ICM830B4.STC05095.D0000107.? CL(JESSPOOL)
807 INSUFFICIENT ACCESS AUTHORITY
807 FROM *.* (G)
807 ACCESS INTENT(UPDATE ) ACCESS ALLOWED(READ )
16.32.58 STC05095 $HASP708 ICM830B4 DSNTRACE OPEN FAILED
808 RC=11 AUTHORIZATION FAILURE
808 DSNNAME=ICMST.ICM830B4.STC05095.D0000107.?
16.32.58 STC05095 IEC150I 913-74, IGG0199G, ICM830B4, ICM830B4, DSNTRACE
16.32.58 STC05095 ICH422I THE ENVIRONMENT CANNOT BECOME UNCONTROLLED.
16.32.58 STC05095 BPXP014I ENVIRONMENT MUST REMAIN CONTROLLED FOR SERVER
(BPX.SERVER) PROCESSING.
16.32.58 STC05095 CSV042I REQUESTED MODULE IDIXDCAP NOT ACCESSED. THE MODULE
IS NOT PROGRAM CONTROLLED
16.32.58 STC05095 IEA995I SYMPTOM DUMP OUTPUT 813
```

```

813      USER COMPLETION CODE=4039 REASON CODE=00000000
813      TIME=16.32.58 SEQ=12404 CPU=0000 ASID=0235
813      PSW AT TIME OF ERROR 078D1400 8526B29E ILC 2 INTC 0D
813      NO ACTIVE MODULE FOUND
813      NAME=UNKNOWN
813      DATA AT PSW 0526B298 - 00181610 0A0D58D0 D00498EC
813      AR/GR 0: 80CD7B3E/07043400_84000000 1:
00000000/00000000_84000FC7

```

---

This error does not occur until the first user tries to access Resource Manager, for example, browsing a document. This error will cause the HTTP server to terminate.

### 13.7.2 Authority to load and execute the CGI program in HTTP server

In addition to the DB2-related discussion, you also have to consider who is loading and executing the Resource Manager CGI program.

It is the user ID that the HTTP server procedure is executing under who needs the EXECUTE privilege on the CGI program `icmmosct.so`. The “UserId” setting in the HTTP configuration file does not have influence on this.

If the started task user assigned to the HTTP server is not authorized to execute the CGI program based on the security settings for the folder where you installed the program, you will experience a program failure during the startup of your HTTP server.

Example 13-14 shows an example from the job log when such a situation occurs. In the example shown, started task user ICMST has no EXECUTE privilege on the `icmmosct.so` program.

#### *Example 13-14 Error if started task user is not authorized to execute RM CGI program*

---

```

15.44.42 STC05079 ---- WEDNESDAY, 16 MAR 2005 ----
15.44.42 STC05079 IEF695I START ICM830B4 WITH JOBNAME ICM830B4 IS ASSIGNED TO
USER ICMST, GROUP STCGRP
15.44.42 STC05079 $HASP373 ICM830B4 STARTED
15.44.42 STC05079 IEF403I ICM830B4 - STARTED - TIME=15.44.42
15.44.42 STC05079 IMW3534I PID: 84214218 SERVER STARTING
15.49.16      ICH408I USER(ICMST ) GROUP(STCGRP ) NAME(USER FOR ICM
STARTED) 422
422      /MVSG/icm830b4/usr/lpp/icm/V8R3M0/bin/icmmosct.so
422      CL(FSOBJ ) FID(01C8C6E2F0F2F80003090000000A0000)
422      INSUFFICIENT AUTHORITY TO OPEN
422      ACCESS INTENT(--X) ACCESS ALLOWED(NO      --X)
422      EFFECTIVE UID(0000000000) EFFECTIVE GID(0000000104)

```

---



The error message will occur at startup of the HTTP server, but it will continue to run. If you glance at the list of running tasks, you may think Resource Manager is active.

Users trying to access Resource Manager will experience a number of different errors, though they can access Library Server functions such as searching, they will fail when trying to either import (see Example 13-15) or display (see Example 13-16) a document.

---

*Example 13-15 Windows client error when importing a document and RM is not active*

---

The system could not create a document object.  
Error details: DGL3854A: Internal error occurred

---

---

*Example 13-16 Windows client error when browsing document and RM is not active*

---

An error occurred in loading the file. The system cannot display it.  
Error details: ::HTTP/1.1 500 Internal Server Error

---

Additional messages can be found in the httpd-errors log (see Example 13-17). These messages are logged when a user tries to access Resource Manager . No further messages are written to the job log.

---

*Example 13-17 Message in httpd-errors log*

---

[16/mar/2005:14:58:51 +0000] [IMW0193I OK] [host: 9.150.143.126] Service handler performed no action; contact the server administrator

---

### 13.7.3 Invalid RM user ID or password in LS configuration

As discussed in previous sections, you define a user ID and password for each Resource Manager defined to Library Server. The user ID is specified during the initial load of the Library Server tables (install job ICMMLSLD) and the user ID and password can be updated using the system administration client afterwards.

If you have specified the “UserId %%CLIENT%%” parameter in the HTTP configuration file, an incorrect user ID or an invalid password will cause access to Resource Manager to fail.

Using the Windows client, the user will experience the errors when displaying a document (see Example 13-18) or import a document (see Example 13-19).

---

*Example 13-18 Windows client error trying to display a document if RM user access fails*

---

An error occurred in loading the file. The system cannot display it.  
Error details: ::HTTP/1.1 401 Unauthorized

---

*Example 13-19 Windows client error trying to import a document if RM user access fails*

---

The system could not create a document object.  
Error details: DGL3854A: Internal error occurred

---

Discovering the cause of this problem may not be straightforward, since you will see no error messages in the HTTP server's job log or the system log. Even starting the detailed Resource Manager trace provided no hints, because the error prevents any Resource Manager activity, including writing trace information, from happening.

The only place you may find a hint is in the httpd-errors log as illustrated in Example 13-20.

*Example 13-20 Message in httpd-errors log if RM user access fails*

---

```
[16/mar/2005:16:20:52 +0000] [IMW0196I NOT AUTHENTICATED] [host: 9.150.143.126]
/ICMResourceManager
```

---

A variation to the invalid password scenario above is shown in Example 13-21. In this example, the Resource Manager user has been revoked due to excessive access attempts.

*Example 13-21 Message in httpd-errors log if RM user access is revoked*

---

```
[16/mar/2005:16:40:49 +0000] [IMW0193I OK] [host: 9.150.143.126]
IMW0573I username=ICMRM; pwlen=4; newpwlen=0; errno=163;
errno2=090c081c;
EDC5163I SAF/RACF extract error.
[16/mar/2005:16:40:49 +0000] [IMW0196I NOT AUTHENTICATED]
[host: 9.150.143.126] /ICMResourceManager
```

---

Example 13-22 shows the error message when the password of the Resource Manager user has expired. As you can imagine, this situation could occur if you would allow the Resource Manager user's password to expire without updating the settings in Library Server. It could take you a little time one Monday morning to figure out what the cause of your Resource Manager problem is if you do not know in which log file to look.

*Example 13-22 Message in httpd-errors log if RM user password is expired*

---

```
[17/mar/2005:11:27:49 +0000] [IMW0193I OK] [host: 9.146.177.47]
IMW0573I username=ICMRM; pwlen=6; newpwlen=0; errno=168;
errno2=090c0000;
EDC5168I Password has expired.
[17/mar/2005:11:27:49 +0000] [IMW0415E PASSWORD EXPIRED]
[host: 9.146.177.47] /ICMResourceManager
```

---



## Part 4

# Appendixes





# WebSphere EAR file creation

This appendix provides you with detailed steps to create an IBM WebSphere Application EAR file by using the IBM WebSphere Application Development Studio.

We use the application developed in Chapter 3, “Content Manager Toolkit for z/OS” on page 45 and show you how to create the EAR file for it.

In Appendix B, “WebSphere application setup” on page 355, we show you the steps to install this EAR file in WebSphere of z/OS.

## Prerequisites to create the EAR file

The following software is required to create a WebSphere EAR file:

- IBM WebSphere Application Development Studio

In this chapter, we do not describe the physical installation or configuration of this software. Please refer to the product installation and configuration documentation.

This software is client-based and is only available for multiplatforms.

## Purpose of the EAR file

The EAR file contains your Web application and its configuration. The application can contain Web pages and it can also contain compiled Java code.

The EAR file also contains a file named `web.xml`. This file contains the names of your Web servlets.

A Web servlet in WebSphere is used when you want to associate a HTTP request with a Java class.

In our example, as provided in Chapter 3, “Content Manager Toolkit for z/OS” on page 45, we have a HTML file. This file contains a Web form that has an action linked to it. The action is in the form of a URL. In the example, it is:

```
http://server:port/root/SearchItemType
```

Where:

- Server is the host name of the system where WebSphere is installed.
- Port is the port number of your Web application configured in Appendix B, “WebSphere application setup” on page 355.
- Root is the context root of your Web application defined in your EAR file.
- SearchItemType is the name of your servlet.

The `web.xml` file contains the following two items for every Web servlet:

- The name of your Java class that must be linked to your Web page.
- The URL that links the Web request or Web form to the servlet.

In our example, the following is a sequence of events and actions that takes place:

1. The user completes the Web form and clicks **submit**.

2. The browser sends a get request to the HTTP server containing the variable-value pairs of the fields on the Web form.
3. The HTTP request is then routed to WebSphere and it then creates an instance of your Java class and calls the get method in it.  
Subsequent calls use the same instance of your Web class.
4. Processing is passed to the Java class and it executes the business logic of your application.

In this example, the Java code formats the resulting page by using the print writer of the HTTP response. The response page is then sent back to the browser of the user containing the search result or error message.

## Create the EAR file

We outline the summary of the steps you need to perform to create your Web project and deploy it to an EAR file:

1. Create a Web project.
2. Download and import the library files.
3. Add the HTML and Java files to your project.
4. Add the Java source to your project.
5. Define your Web servlet in the project.
6. Step 9: Create an EAR file.

In the remaining section of this appendix, we describe these steps in detail.

## Create a Web project

To create a Web project, perform the following steps:

1. Launch WebSphere Application Developer Studio.
2. Select **File** → **New** → **Project**.
3. Select **Web** on the left side of the panel, and select **Dynamic Web Project** from the right side of the panel (see Figure A-1).
4. Click **Next**.

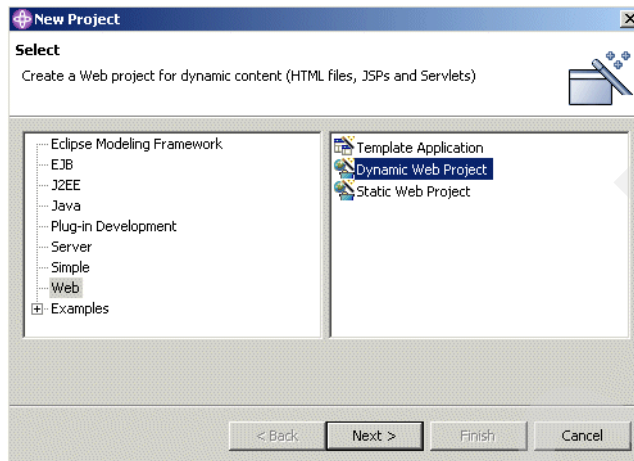


Figure A-1 Create a Web project

5. Enter an unique name for the project, for example, zostest. Check the **Configure advanced options** box (see Figure A-2).
6. Click **Next**.

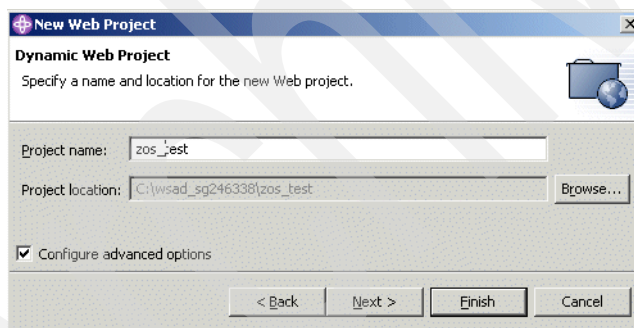


Figure A-2 Provide a project name

7. Specify the J2EE settings (see Figure A-3).



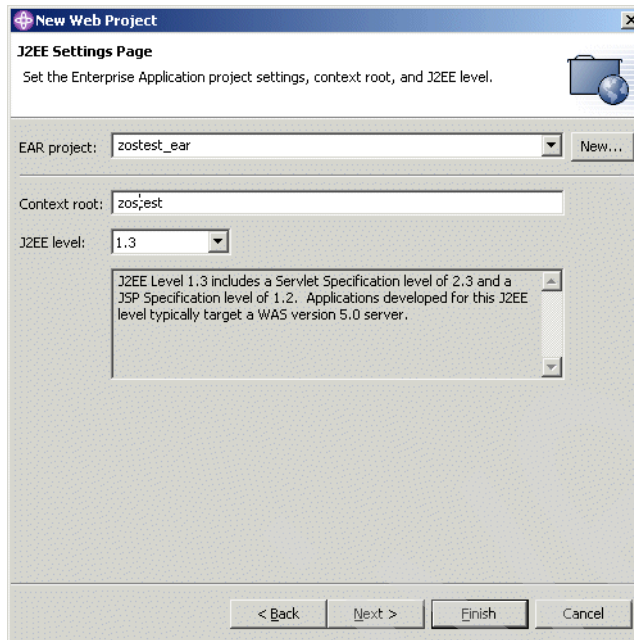


Figure A-3 Specify J2EE settings

You need to perform the following steps:

- a. Click **New** to create a new EAR or deployment project.
- b. On the next screen, enter the name of your EAR project.

In our example, it is `zostest_ear`. After the step, you will be returned back to this page. The name of the EAR project must be different than the name of the Web project.

- c. Enter the name of your context root.

This is the name used for your Web site. It is in the URL between the host name and the HTML file name.

- d. Select **1.3** for J2EE level.

This is the level for Content Manager V8.3. It might differ for future versions or releases.

- e. Click Finish.

You have now created a Web project and a deployment project for it. All the Java code and HTML pages have to be added to the Web project. The EAR project is only used for deployment when you want to create the EAR file to install in your WebSphere application server running on z/OS.

## Download and import the library files

You need to download the JAR files from your mainframe and import them to your Web project.

### Download the library files

Download the library with the following steps:

1. Go to the command prompt on your local workstation.
2. Enter the command **ftp hostname**.
3. Type **cd /path**.
4. Type **lcd /localpath**.
5. Type **bin**.
6. Type **prompt\$**.
7. Type **mget \*.jar**.
8. Type **quit**.
9. Type **exit**.

Where:

- ▶ Hostname is the IP address or host name of the mainframe where the Content Manager toolkit for z/OS is installed.
- ▶ Path is the name of the directory in z/OS where the toolkit was installed.
- ▶ Localpath is any path on your local workstation that you have to create before the download.

The following is the list of JAR files that will be in the local path after the download:

```
cmb81.jar  
cmbcm81.jar  
cmbicm81.jar  
cmblog4j81.jar  
cmbstdk81.jar  
cmbutil81.jar  
cmbutilicm81.jar  
icmrm81.jar  
log4j-1.2.8.jar  
xerces.jar
```

Note that these files are only included in the Web project to correct compilation errors. They will not be part of the application EAR file.

**Note:** The external libraries are for z/OS and they cannot be used to execute or test your Web application in the WebSphere Application Development Studio.

You can use the Information Integrator for Content for Multiplatforms to debug your Web application.

## Import the library files

You need to add the library files you just downloaded as the external JAR files to your project (see Figure A-4).

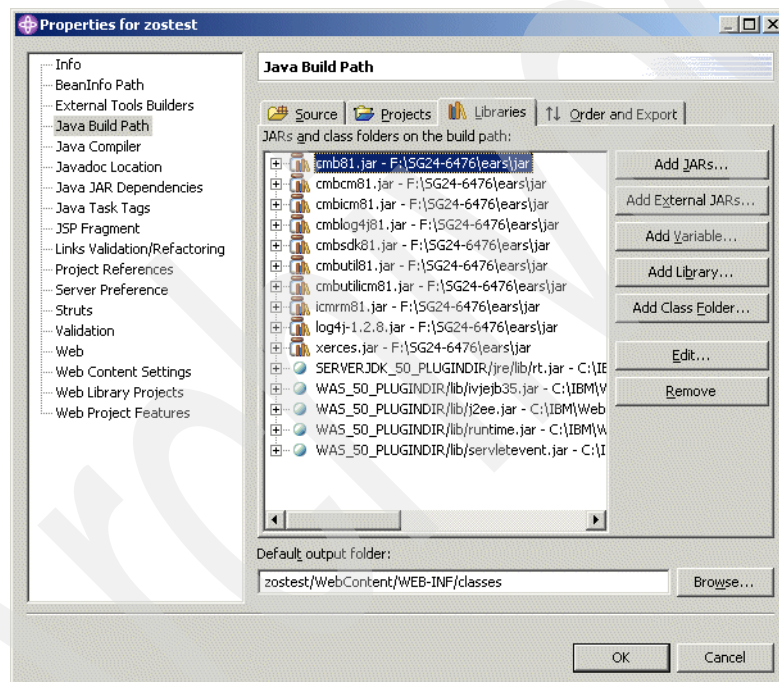


Figure A-4 Import external libraries

To do that, perform the following steps:

1. Right-click **zostest** and select properties.
2. Select **Java build path** from the left side of the panel.
3. Click **Add External JARs....**
4. Add the following files:

cmb81.jar

cmbcm81.jar  
cmbicm81.jar  
cmblog4j81.jar  
cmbsdk81.jar  
cmbutil81.jar  
cmbutilicm81.jar  
icmrm81.jar  
log4j-1.2.8.jar  
xerces.jar

Figure A-4 shows the screen where you can select the external JAR file and add it to your project.

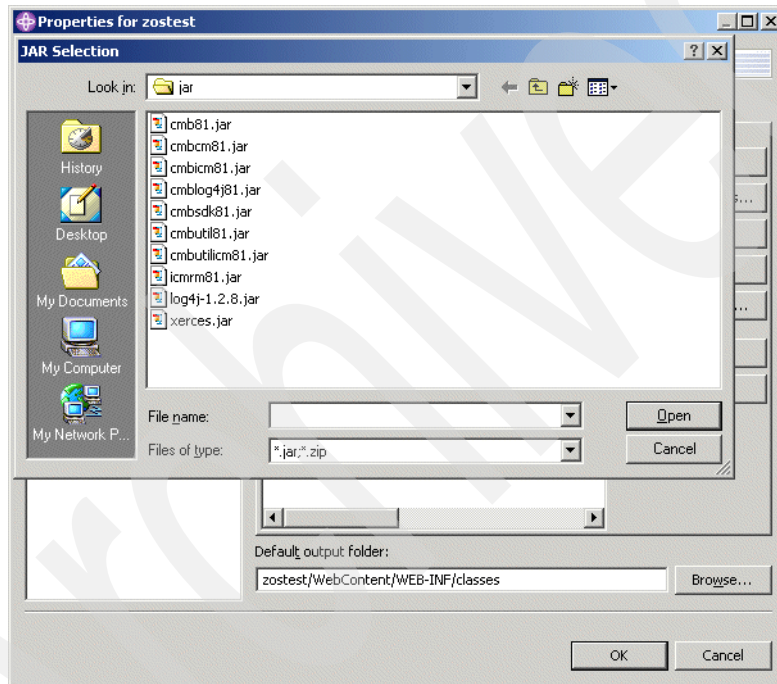


Figure 1 Add external JAR file

Perform the following steps to add an external JAR file to your project:

- Use the look in navigation combo box to select the local path on your computer where the JAR file is located.
- Select the relevant JAR file.
- Click **Open**.

Remember to repeat these steps for every JAR file.

5. Once you have imported all the external JAR files, click **OK** to save the Java build path.

**Note:** Do not remove any of the library files that were already added to your Web project.

## Add the HTML and Java files to your project

You now need to add the HTML and Java files to your project.

### Add HTML file

Figure A-5 shows where you add your HTML file.

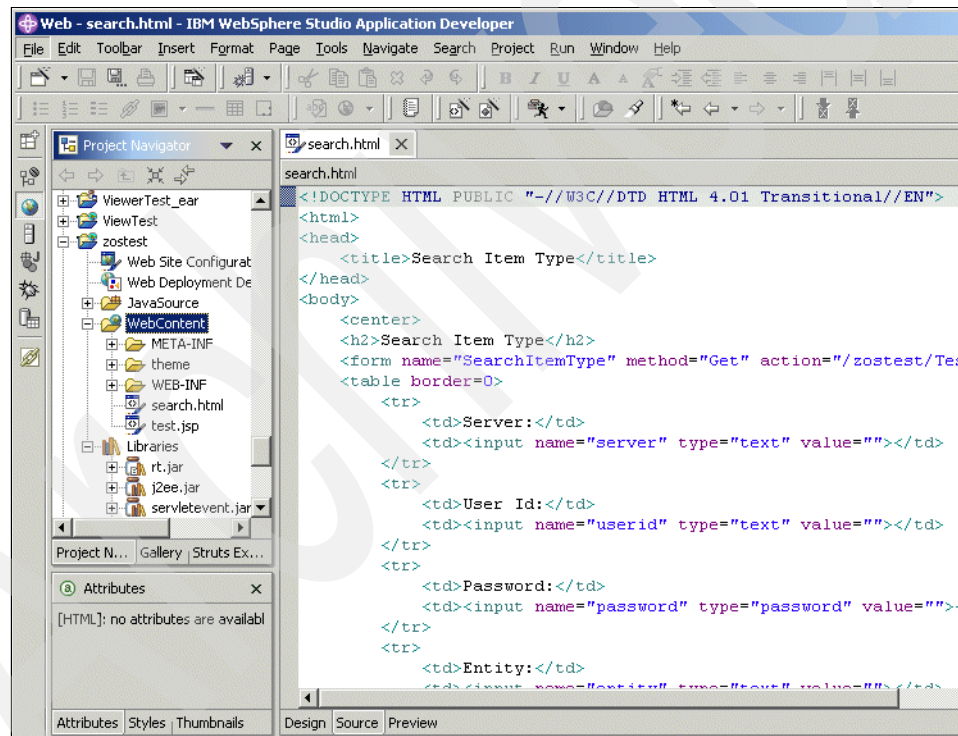


Figure A-5 Add HTML file to the project

Do the following steps to add the HTML file to your project:

1. Click **zostest**.
2. Right-click **Web content** and select **New** → **HTML/XHTML** file.
3. Enter the name of the HTML file. In this example, it is search.html.

4. Click **Finish**.
5. Enter the HTML code in the right panel as described in Chapter 3, “Content Manager Toolkit for z/OS” on page 45.
6. Select **File** → **Save**.

The name of the Web page or HTML file can be anything. This is the starting point when you want to test your Web application with the URL:

`http://host_name:websphere_port/content_root/web_page.html`

## Add the Java source to your project

Now you are ready to add the Java source to your project. See Figure A-6.

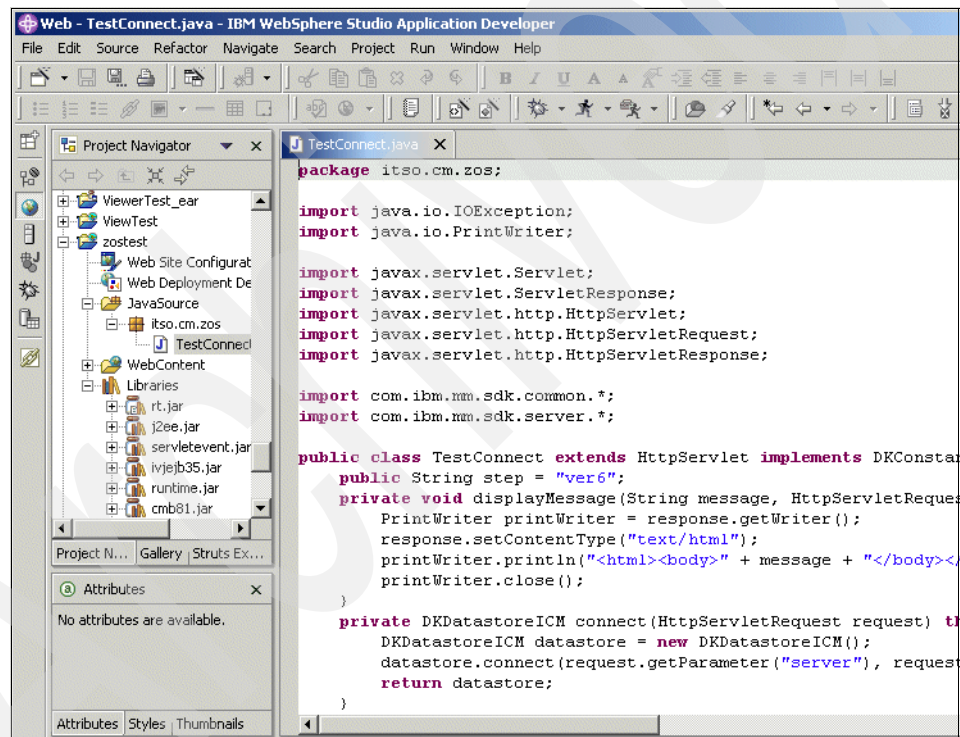


Figure A-6 Add Java source code in the Web project

Perform the following steps to add the Java source to your project:

1. Click **zostest**.
2. Right-click **Java source**, and select **New** → **Package**.
3. Enter the name of the package. In this example, it is `itso.cm.zos`.



4. Click **Finish**.
5. Right-click **Java source**, and select **New** → **Class**.
6. Enter the name of the class. In this example, it is **TestConnect**.
7. Click **Finish**.
8. Enter the Java code in the right panel as described in Chapter 3, “Content Manager Toolkit for z/OS” on page 45.
9. Select **File** → **Save**.

## Define your Web servlet in the project

Now, it is time to define your Web servlet in the project as shown in Figure A-7.

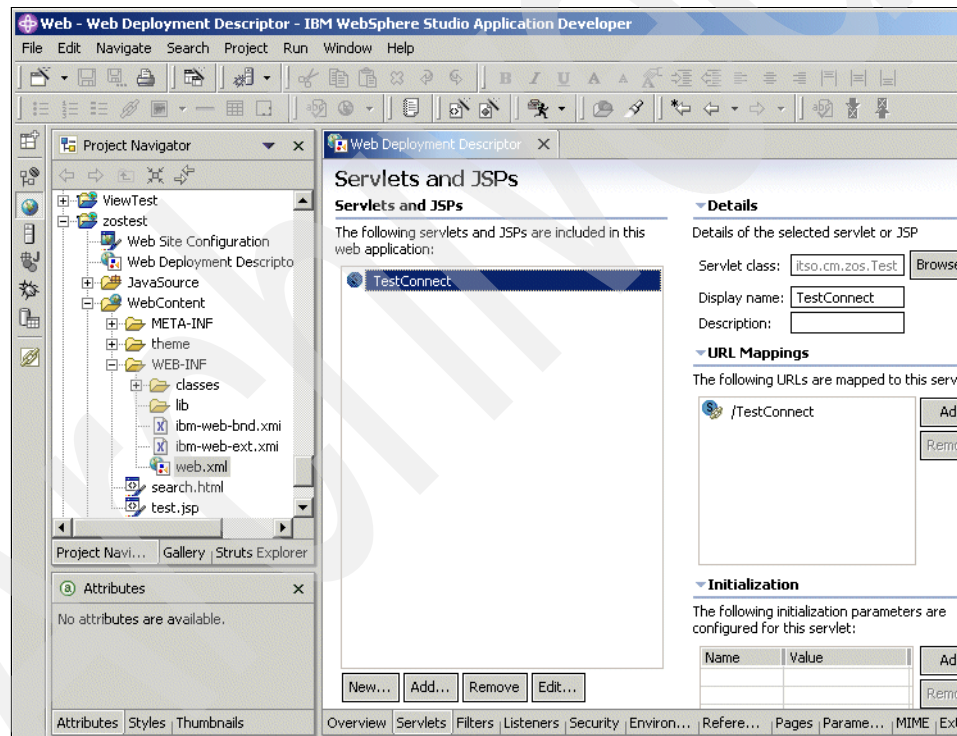


Figure A-7 Define Web servlet in the project

To define your Web servlet in the project, perform the following steps:

1. Click **zostest**.
2. Right-click **Web content** → **WEB-INF**.
3. Double-click **web.xml**.

4. Click the **servlets** tab in the bottom of the right pane.
5. Click **Add**.
6. Click **SearchItemType**.
7. Click **Add** in the URL mappings section.
8. Click **OK**.
9. Select **File** → **Save**.

The name of the servlet can be anything. It is more important that you use the same name for the Java source and the action in the HTML file.

## Step 9: Create an EAR file

Once you have created your project, imported library files, added the HTML file and Java source code, and defined the Web servlet, you can package everything together in an EAR file. See Figure A-8.

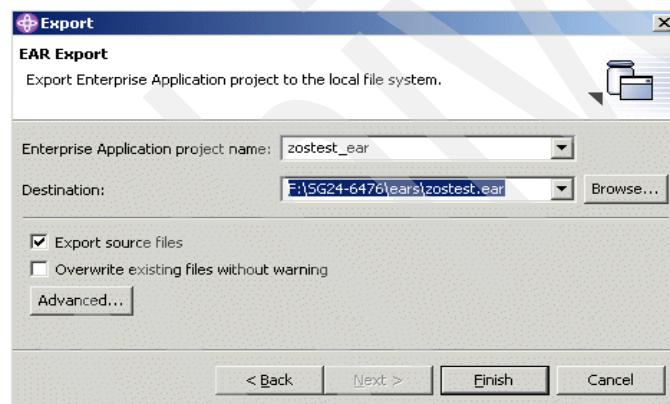


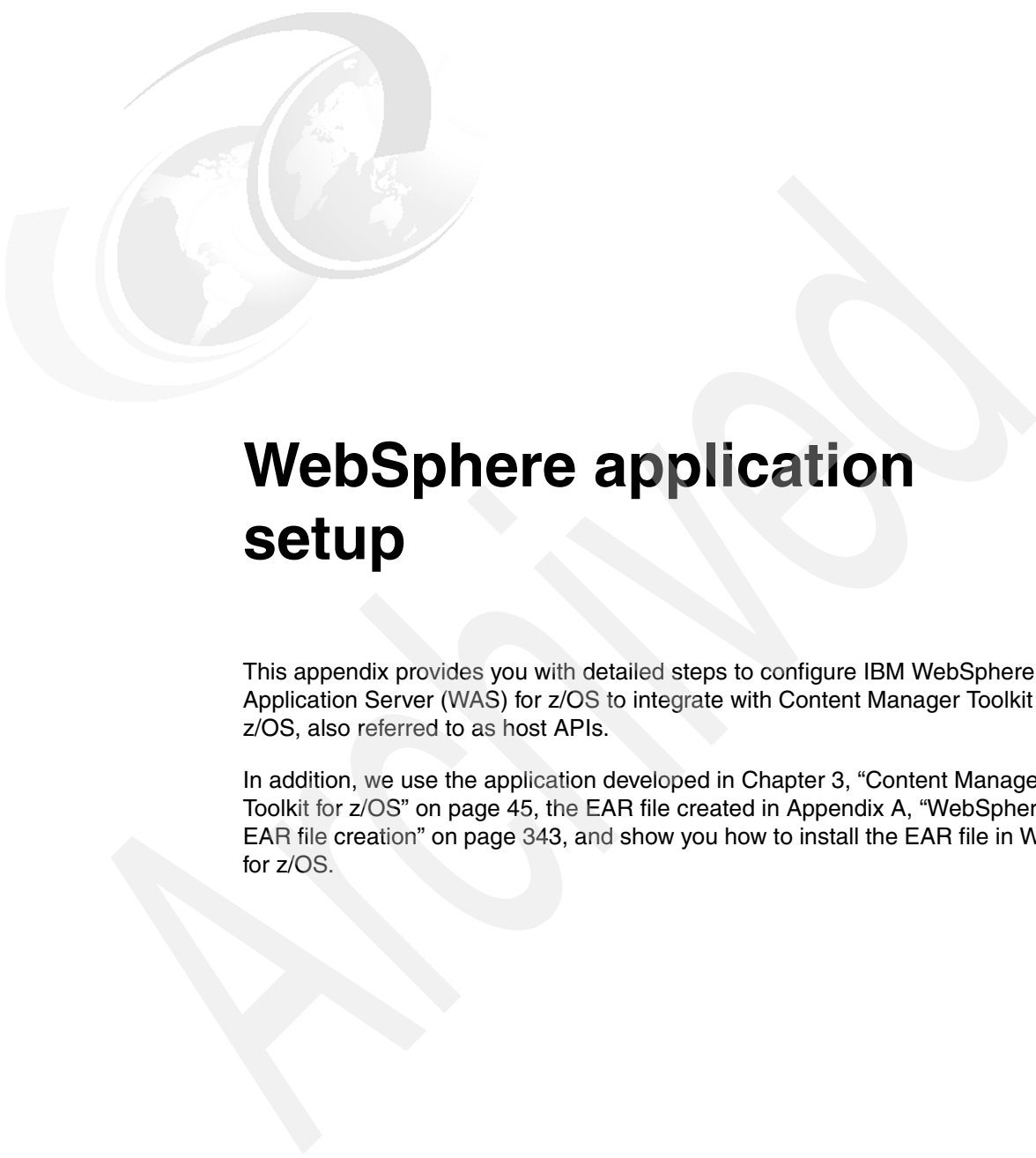
Figure A-8 Create EAR file

Perform the following steps to create your EAR file:

1. Right-click **zostest\_ear** and select **Export** → **EAR**.
2. Click **Next**.
3. Enter the destination path and file name where you want to save the EAR file.
4. Click **Finish**.

You have now created your EAR file. It contains your Web application. You have to install the EAR file in WebSphere Application Server for z/OS. This step is described in detail in Appendix B, “WebSphere application setup” on page 355.





# WebSphere application setup

This appendix provides you with detailed steps to configure IBM WebSphere Application Server (WAS) for z/OS to integrate with Content Manager Toolkit for z/OS, also referred to as host APIs.

In addition, we use the application developed in Chapter 3, “Content Manager Toolkit for z/OS” on page 45, the EAR file created in Appendix A, “WebSphere EAR file creation” on page 343, and show you how to install the EAR file in WAS for z/OS.

## Prerequisites for WebSphere integration

The following software is required for the integration of WebSphere Application Server for z/OS and Content Manager. This software must be fully configured and operational:

- ▶ IBM DB2 Content Manager for z/OS
- ▶ IBM DB2 Content Manager Toolkit for z/OS
- ▶ IBM WebSphere Application Server for z/OS
- ▶ IBM HTTP Server for z/OS
- ▶ IBM DB2 JDBC or JCC for z/OS

In this chapter, we do not describe the physical installation or configuration of this software. For instructions, refer to the installation or configuration documentation.

## Purpose of WebSphere integration

You use WAS to create a Web application. This can be HTML pages that may contain forms that are linked to Java servlets. The Java code in the servlet can then call the APIs from Content Manager Toolkit or z/OS.

In this chapter we show you how to configure the Web application developed in chapter 2 of this document.

You need to create an EAR file before you can configure the Web application. This is described in appendix 2 of this document.

## Components of WebSphere integration

The WebSphere Application Server (WAS) integration has the following components:

- ▶ The WAS enterprise server that is used as a container or an execution environment for the WAS application.
- ▶ The WAS enterprise application that contains the Java code that has to be executed and the Web pages that will be displayed.
- ▶ The HTTP server that is used to link the requests from the browsers of the users to the application running on the WAS server.

# WebSphere configuration

The WebSphere configuration and setup is done from its administrative console.

The following is a summary of steps that are involved:

1. WAS enterprise server configuration:
  - a. Create a new WebSphere application server.
  - b. Configure the new WebSphere application server.
2. WAS enterprise application configuration:
  - Create a new WebSphere application.
3. WAS HTTP server configuration:
  - a. Define the virtual host.
  - b. Update the WebSphere server plugin.

## Log in to WebSphere administrative console

Enter the following URL from a browser to get to the administrative console:

[http://was\\_server:9080/admin](http://was_server:9080/admin)

Where:

- ▶ was\_server is the host name of your WebSphere server.
- ▶ 9080 is the default administration port number.  
Your port number may be different.

Figure B-1 shows the login to the WAS administrative console.

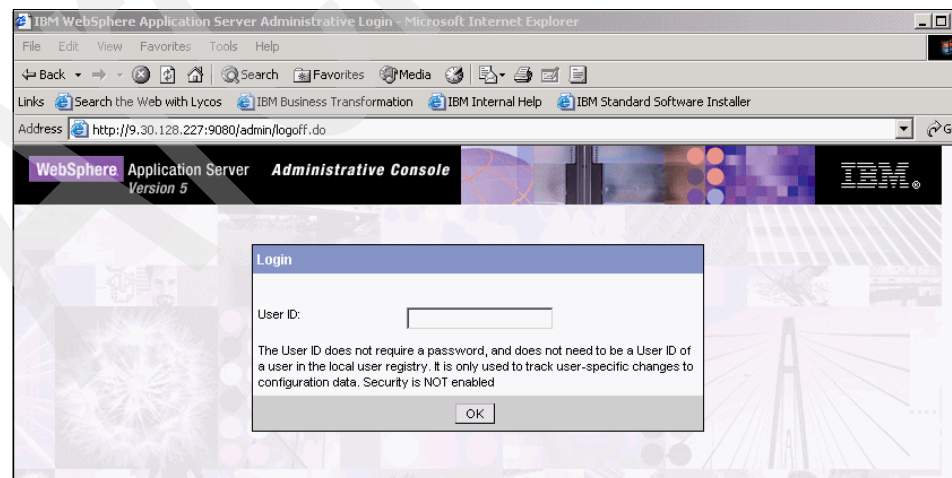


Figure B-1 WebSphere Application Server administrative console login

You may use any user ID if WAS security is not enabled. If it is enabled, you need to contact your WAS administrator for a valid user ID and password.

## WAS enterprise server configuration

Once you are in the WAS administrative console, you are ready to create and configure a new WebSphere application server.

### Create a new WebSphere application server

Perform the following steps to create a new application server:

1. From the left side panel, click **Servers** to expand the list of servers.
2. Click **Application Servers** to expand the list of application servers.  
Figure B-2 appears.

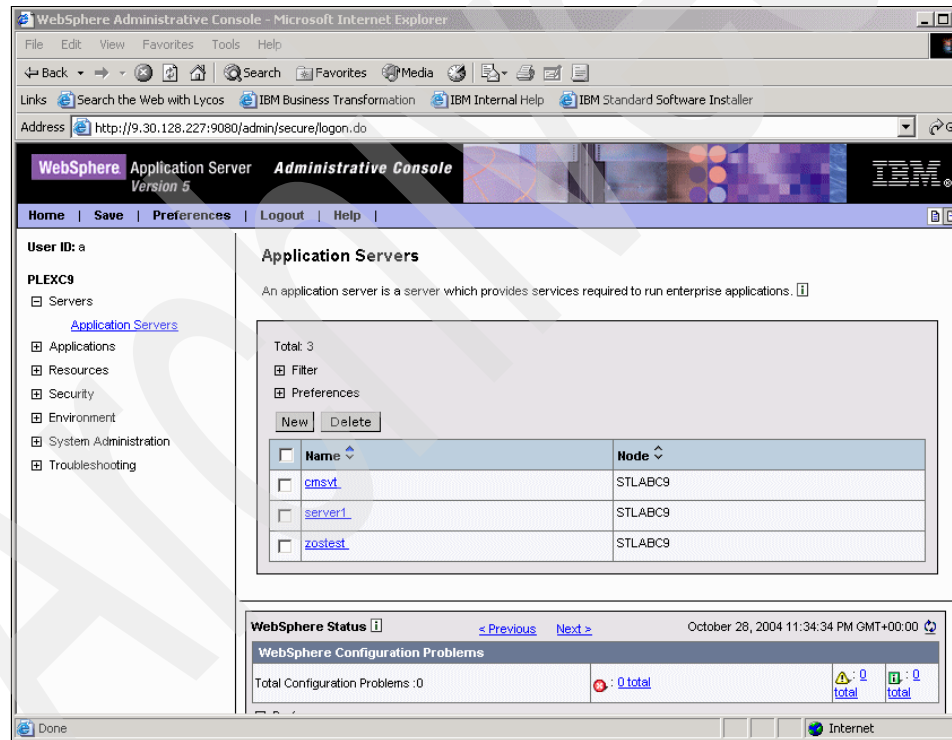


Figure B-2 WAS application server setup

3. Click **New**, from the right side panel, to create a new application server.  
Figure B-3 appears.

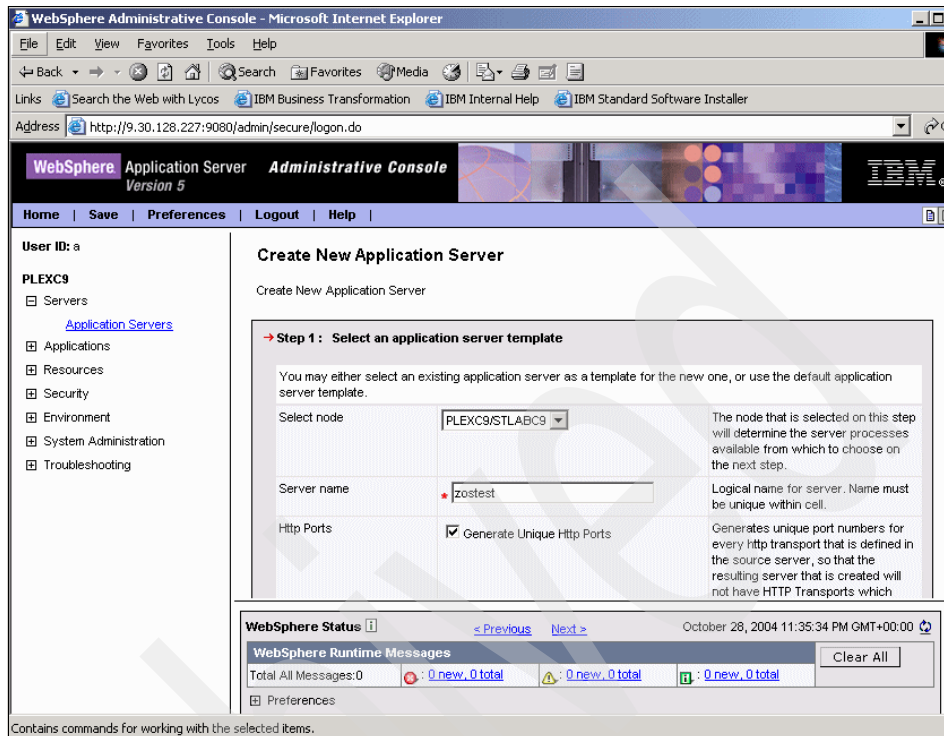


Figure B-3 New application server setup

4. Select an application server template.

In this example, we use the following server name: zostest.

**Note:** The name of the server is case-sensitive.

5. For other fields, use default values provided by WebSphere and click **Next**.

Figure B-4 shows the next screen where you confirm the setup.

6. Confirm the setup, and click **Finish** to create the new application server.

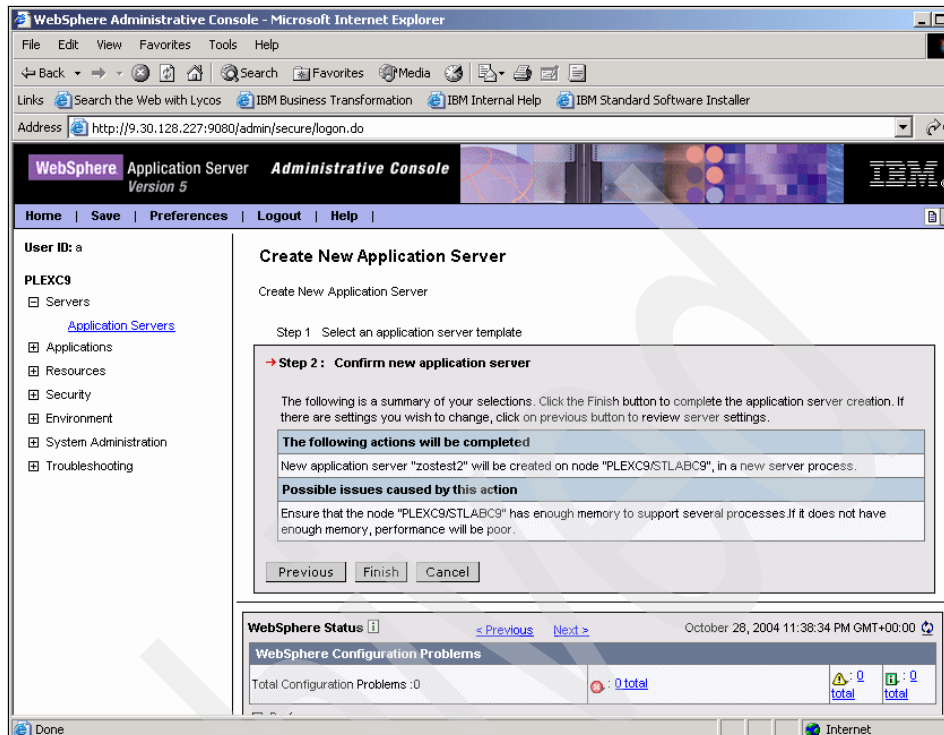


Figure B-4 Confirm new application server setup

## Configure the new WebSphere application server

Once the application server is created, you need to configure it with the following steps from the WebSphere administrative console:

1. From the left side panel, select **Servers** → **Application Servers** → **zostest** where **zostest** is the application server we just created.
2. Click **Process Definition** → **Servant** → **Java Virtual Machine**. Figure B-5 appears.

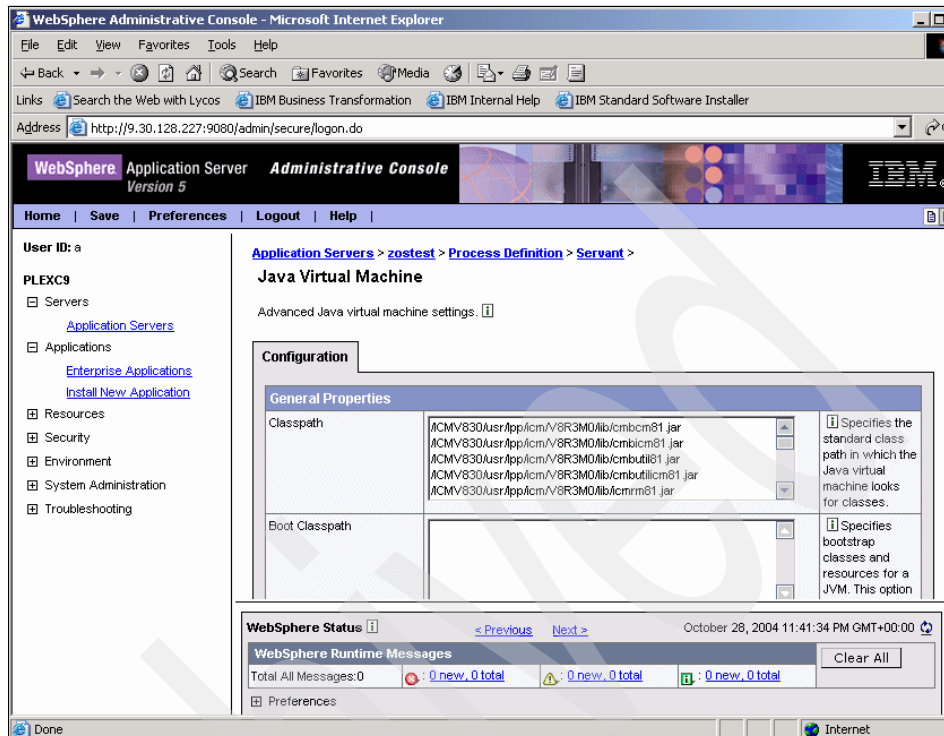


Figure B-5 Java virtual machine setup

### 3. Add the class path of Content Manager Toolkit for z/OS.

The following lines must be added to the class path:

```
/usr/lpp/icm/V8R3M0/lib/cmbcm81.jar
/usr/lpp/icm/V8R3M0/lib/cmbicm81.jar
/usr/lpp/icm/V8R3M0/lib/cmbutil81.jar
/usr/lpp/icm/V8R3M0/lib/cmbutilicm81.jar
/usr/lpp/icm/V8R3M0/lib/icmrm81.jar
/usr/lpp/icm/V8R3M0/lib/cmbjdbcd81.jar
/usr/lpp/icm/V8R3M0/lib/cmb81.jar
/usr/lpp/icm/V8R3M0/lib/cmbstdk81.jar
/usr/lpp/icm/V8R3M0/lib/log4j-1.2.8.jar
/usr/lpp/icm/V8R3M0/lib/cmblog4j81.jar
/usr/lpp/icm/V8R3M0/lib/xerces.jar
/usr/lpp/icm/V8R3M0/cmgmt
/usr/lpp/db2/db2710/jcc/classes/db2jcc.jar
/usr/lpp/db2/db2710/jcc/classes/db2jcc_javax.jar
/usr/lpp/db2/db2710/jcc/classes/db2jcc_license_cisuz.jar
```

### 4. Click **Apply**.

## WAS enterprise application configuration

Once you created and configured the new WebSphere application server, you are ready to create and configure a new WebSphere enterprise application.

### Create a new WebSphere application

To create a new WebSphere enterprise application, perform the following steps:

1. From the WebSphere administrative console, click **Applications** → **Install New Applications**. Figure B-6 appears.

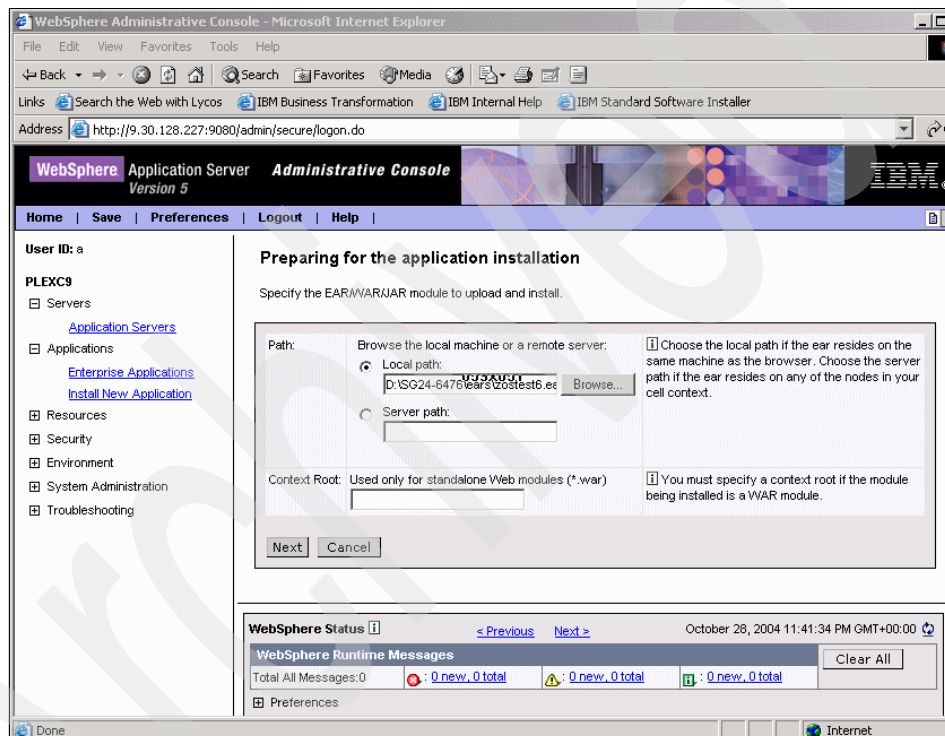


Figure B-6 New application installation preparation

2. Enter the location of the EAR file on your local computer and click **Next**. Figure B-7 appears.

If you have not created your EAR file yet, refer to Appendix A, “WebSphere EAR file creation” on page 343 to create it.



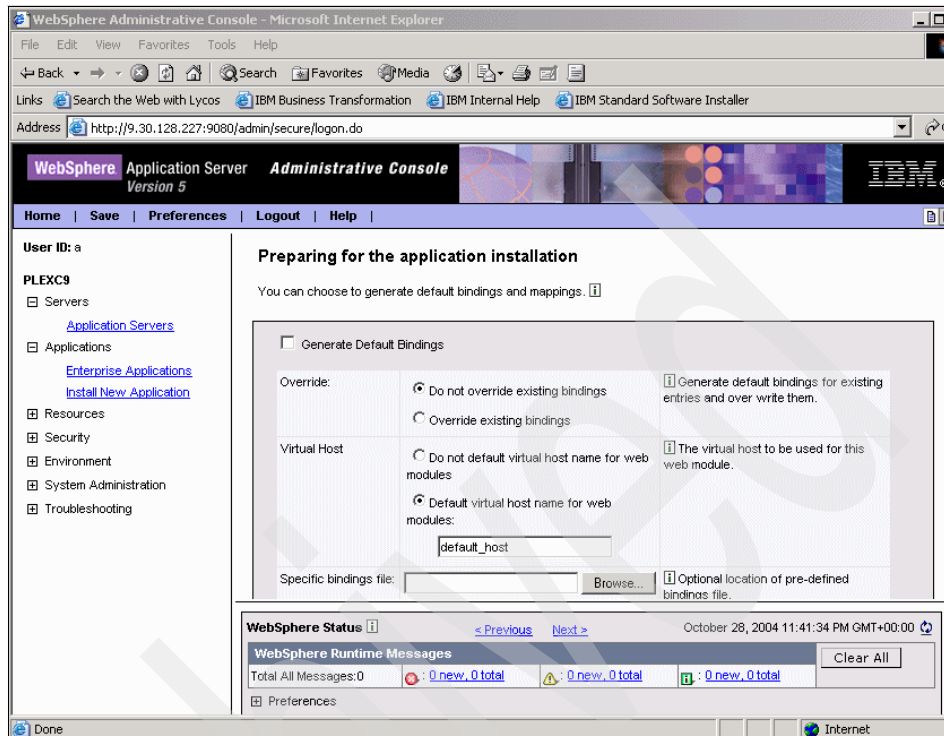


Figure B-7 New application installation preparation: Bind app to virtual host

Chapter 3, “Content Manager Toolkit for z/OS” on page 45 contains the source code of the HTML Web page and the Java servlet. It also describes how the example works.

3. This is where you bind the application to the default virtual Web module. Use the default settings and click **Next**. Figure B-8 appears.

Refer to the next section of this appendix on how to configure the default virtual host Web module.

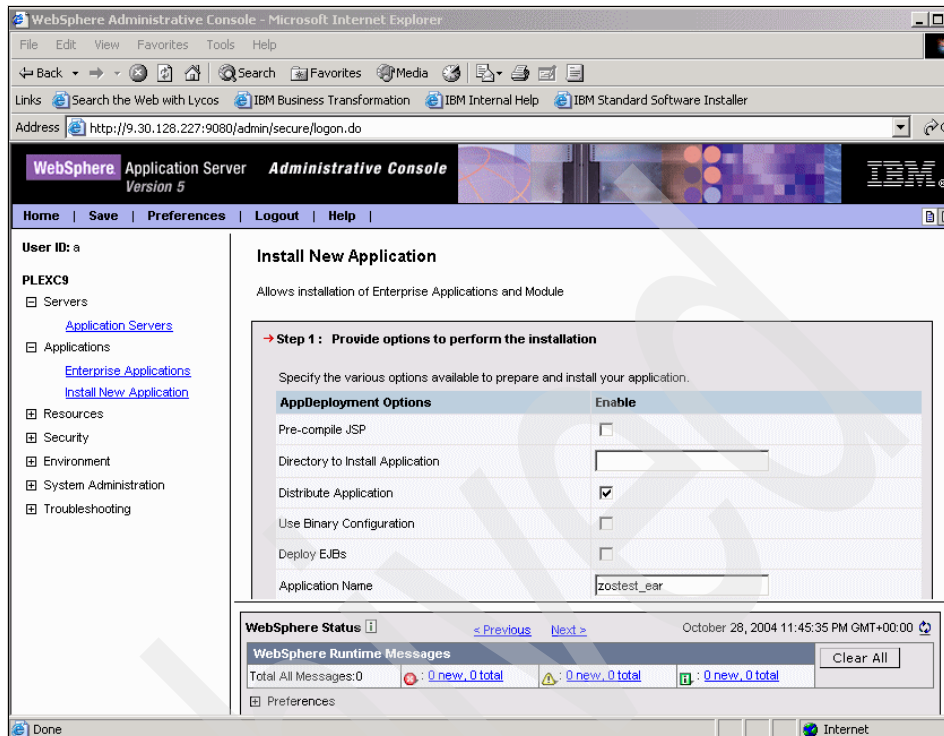


Figure B-8 New application installation setup

4. This is where you install the modules of the enterprise application. Use all the default settings that are contained in the EAR file and click **Next**.

Note, it is better to alter these values in your EAR file and not on this page. Figure B-9 appears.

Refer to the next section of this appendix on how to configure the default host Web host.

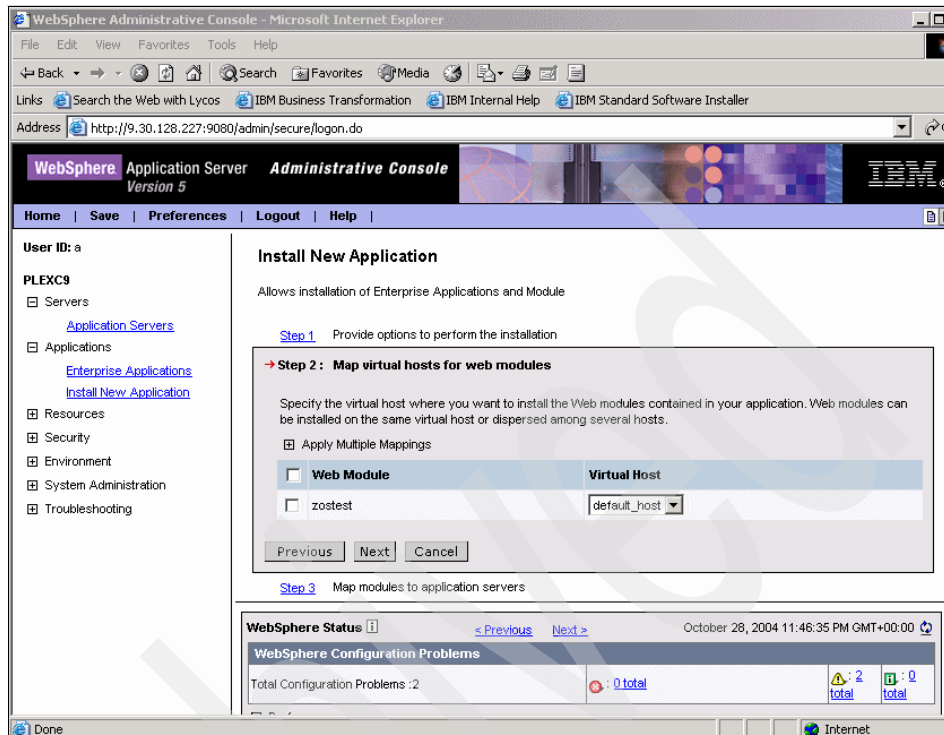


Figure B-9 New application installation setup: Map Web modules to virtual host

5. This is where you map your Web modules to a virtual host defined in WebSphere. Click **Next** and Figure B-10 appears.

Refer to the previous section in this appendix on how to configure a new server in WAS.

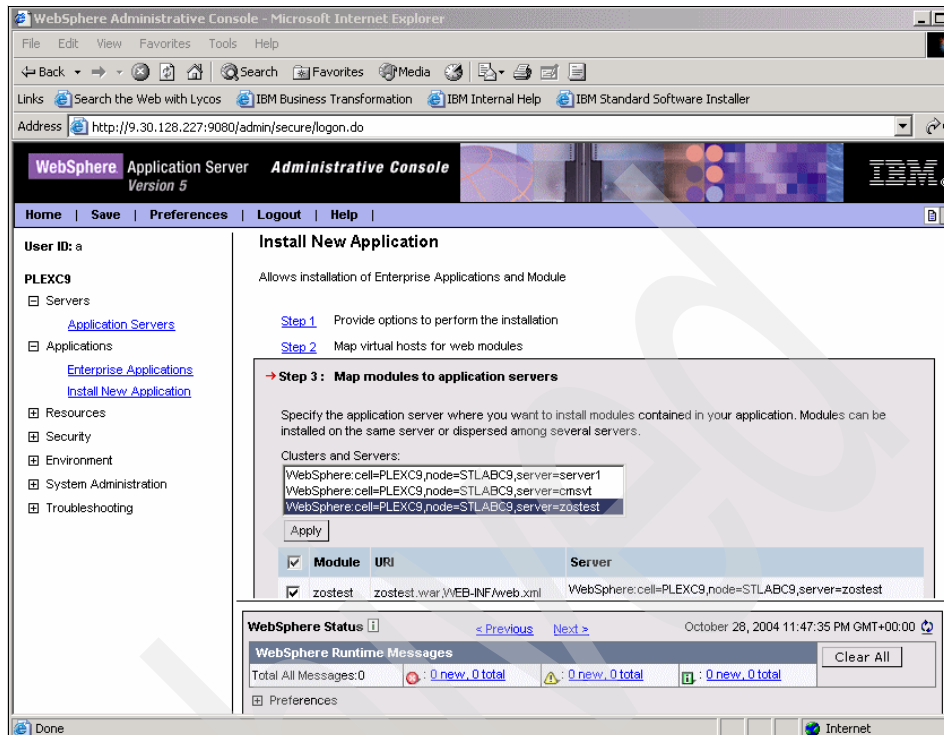


Figure B-10 New application installation setup: Map application to server

6. This is where you map your enterprise application to an enterprise server in WebSphere. Perform the following steps:
  - a. Select the check box next to the module **zostest**.
  - b. From the Clusters and Servers selection list, select the entry that contains **server=zostest**.
  - c. Click **Apply** and then click **Next**. Figure B-11 appears.

**Important:** Confirm that the server of the module **zostest** changes from **server1** to **zostest** after you click **Apply**. If this is not done correctly, the application will be installed on **server1**.

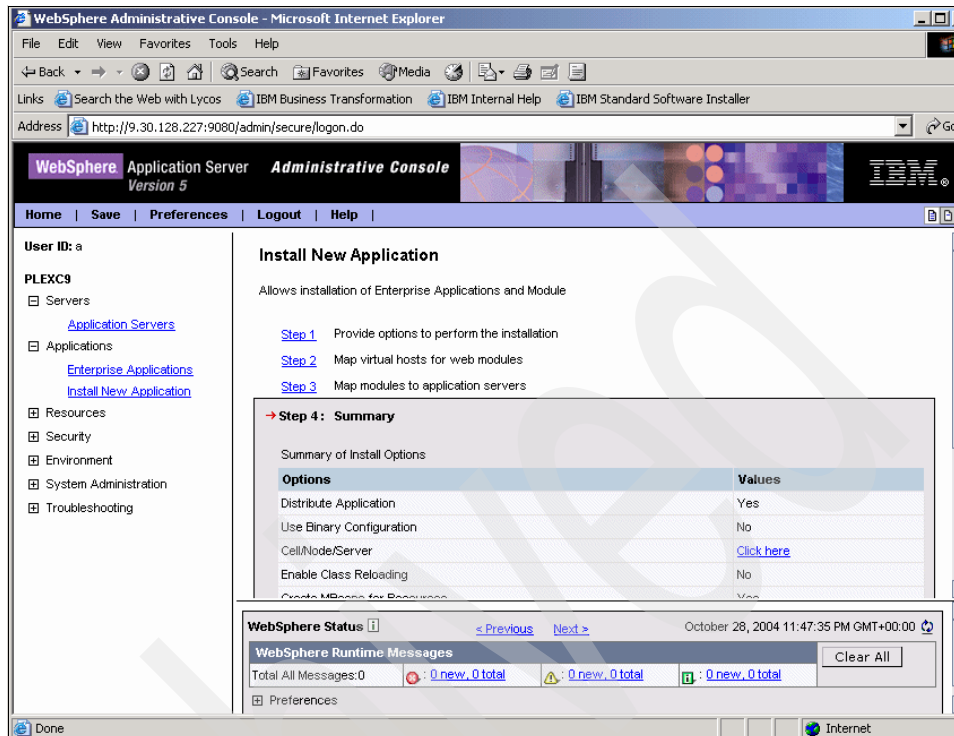


Figure B-11 New application installation: Summary

- Review the summary screen, and click **Finish** to create the new application. Figure B-12 appears.

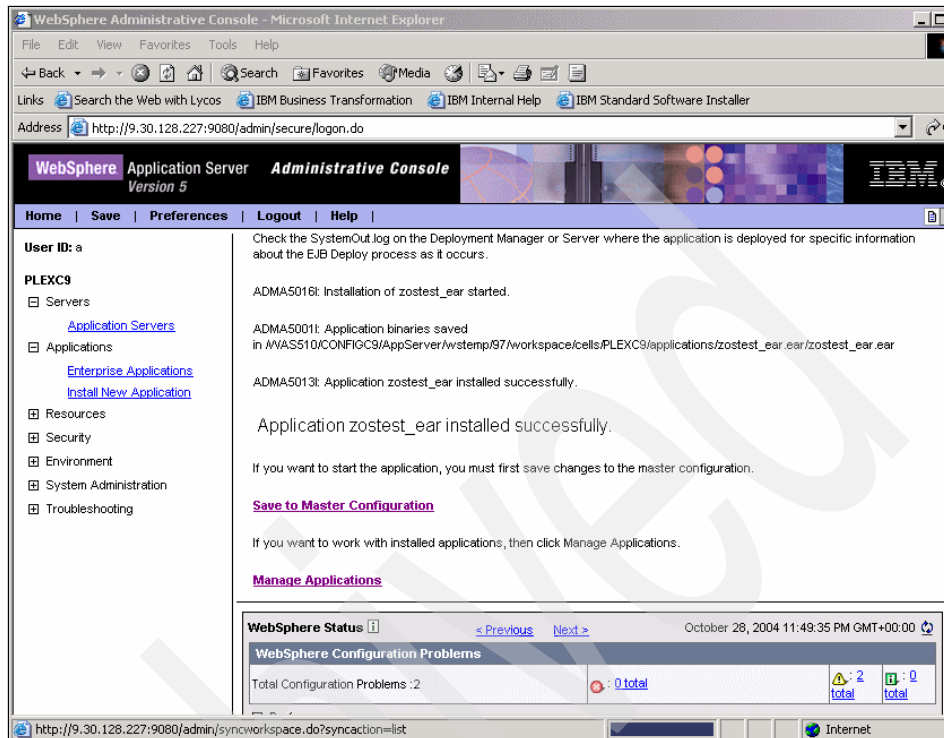


Figure B-12 New application installation: Save to master configuration

8. Click **Save to Master Configuration**. Figure B-13 appears.

**Important:** You must do this step, or else you will lose the definition of the server and the application.

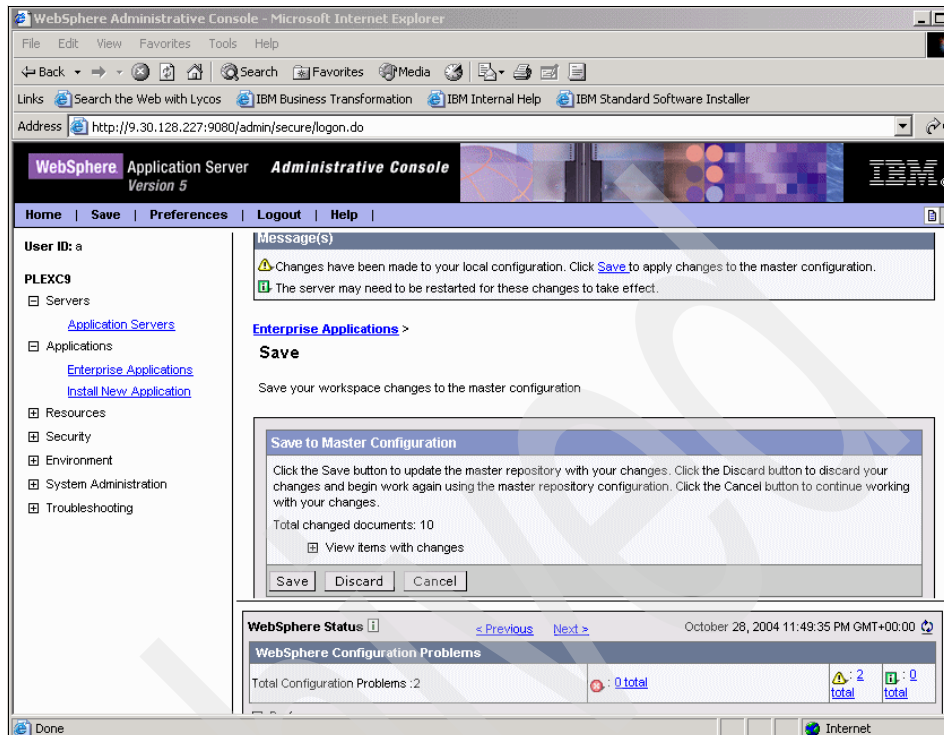


Figure B-13 New application installation: Save

9. Click **Save**.

## WAS HTTP server configuration

The last step is to configure the HTTP server. You need to define the virtual host and update the WebSphere server plugin.

### Define the virtual host

In the WebSphere administrative console, perform the following steps:

1. From the left side panel, click **Environment** → **Virtual Hosts**.
2. Click **Default Host**. Figure B-14 appears.

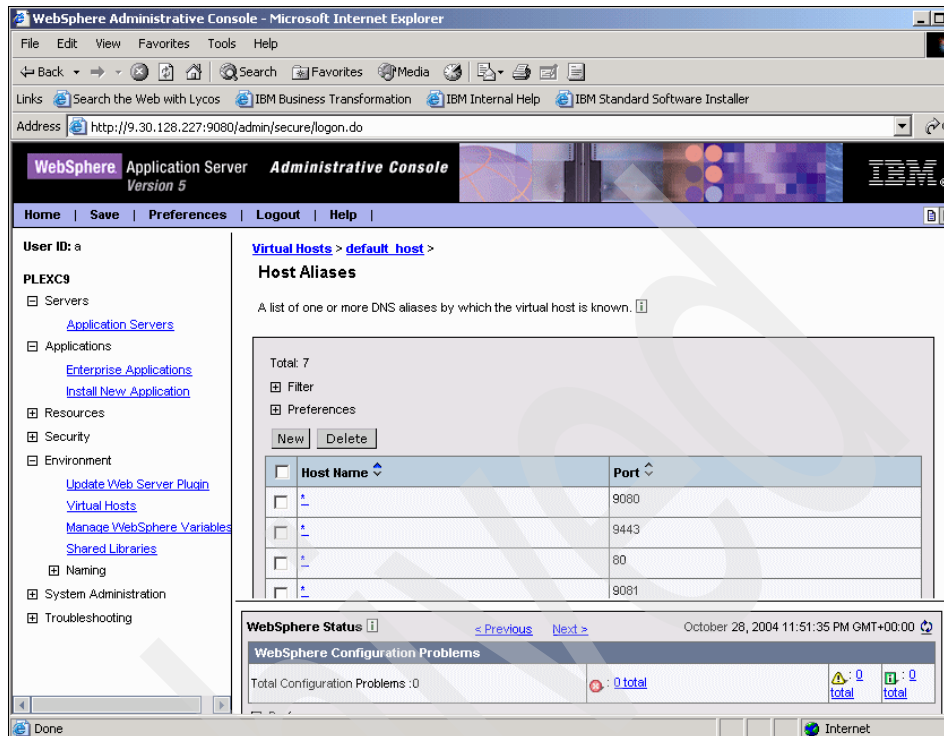


Figure B-14 WAS HTTP server configuration: Host alias

3. This is where you define the virtual and default host, and where you configure the port numbers of your Web application. Click **New**.
4. Enter \* for the host name, and enter the port number that your application will use.
5. Click **Apply** and then click **Save**.

This step is optional when you use a port number that is already defined.

## Update the WebSphere server plugin

To update the WebSphere server plugin configuration, perform the following steps in the WAS administrative console:

1. From the left side panel, click **Environment** → **Update Web Server Plugin**. Figure B-15 appears.



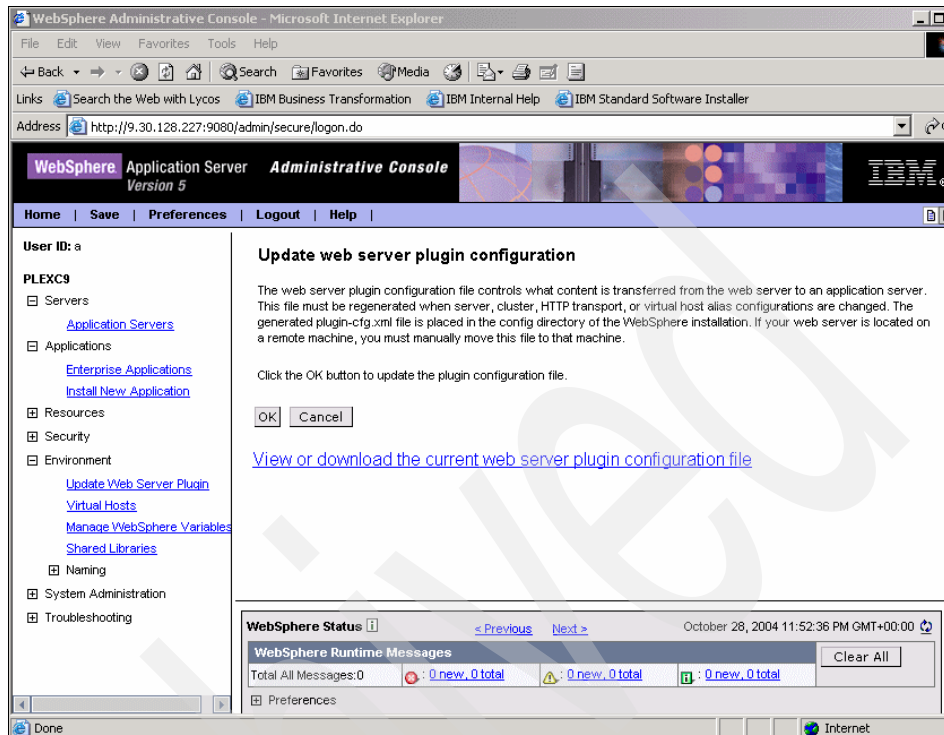


Figure B-15 WAS HTTP server configuration: Update Web server plugin

2. Click **OK** and then click **OK** again.
3. Stop and start the HTTP server.

The plugin file is used to define the WAS configuration in the HTTP server. This enables the HTTP server to pass any HTTP request that is related to the Web application to WebSphere.

Once WebSphere receives the request, it will be executed by the enterprise application containing the Java and HTML source running on the application server.

## Test the application

You may now test your application with the following URL:

<http://server:port/root/SearchItemType.html>

where:

- ▶ Server is the host name of the system where WebSphere is running.
- ▶ Port is the port number that you have defined in your virtual host.
- ▶ Root is the context root that you have defined in your EAR file.
- ▶ SearchItemType.html is the name of your Web page.

## WAS operational information

The following is useful operational information to help you manage your Web applications.

### Stop the application server

Use the following command to stop your WebSphere application server.

In the Path <WAS\_HOME>/bin, enter the command:

```
stopServer.sh zostest
```

where zostest is your application server in WAS.

### Start the application server

Use the following command to start your WebSphere application server.

In the Path <WAS\_HOME>/bin, enter the command:

```
startServer.sh zostest
```

where zostest is your application server in WAS.

### Schema configuration file

In WebSphere, the APIs read the following Content Manager Toolkit for z/OS configuration file to get the name of the Library Server and DB2 schema:

```
cmgmt/connectors/cmbicmsrvs.ini
```

### Change trace level

You can set the log or trace level operation of Content Manager Toolkit for z/OS in the file:

```
cmblogconfig.properties
```

You can increase the logging level of the APIs by using the following statement:

```
DKLogPriority=TRACE
```

You can also add the following statement to get a low level API trace:

```
DKAPIJavaLogLevel=DEBUG
```

### **View log files**

The log files of your Web applications are located in the following path:

```
<WAS_Home>\logs\zostest
```

where zostest is the name of your application server.

The log files of Content Manager Toolkit for z/OS are written in the path as specified by the file:

```
cmblogconfig.properties
```





## **Sample code to change default collections**

This appendix provides the source code of an example that changes the default collection names of all item types starting with a specified prefix in the name.

## Source code for changing default collections

Example C-1 contains the source code for changing the default collection names of all the item types starting with a specified prefix in the name.

*Example: C-1 Change default collection names with a specified prefix*

```
// Change the default collection names of all item types starting with a prefix
// Usage: java ChangeGroup database user password entity group
// Where database is the name of the Library Server
// and user is a administrative user name
// and password is the password of the user
// and entity is an item type prefix (all item types that start with...)
// and group is the new collection name
// and action can be update or view
// It is best to first run the program with the action view to see what items
// will be updated
import com.ibm.mm.sdk.common.*;
import com.ibm.mm.sdk.server.*;
public class ChangeGroup {
    // main method
    public static void main(String argv[]) throws Exception {
        if (argv.length == 6) {
            String database = argv[0]; // Library Server
            String userName = argv[1]; // user name
            String password = argv[2]; // password
            String entity = argv[3]; // prefix of entity
            String collName = argv[4]; // new collection name
            String action = argv[5]; // action update or view
            // connect to Library Server
            DKDatastoreICM dsICM = new DKDatastoreICM();
            dsICM.connect(database,userName,password,"");
            System.out.println("Connected to datastore (Database '" +
                dsICM.datastoreName() +
                "', UserName '" +
                dsICM.userName()+"').");
            DKDatastoreDefICM dsDefICM = (DKDatastoreDefICM)
dsICM.datastoreDef();

            // get collection id of collection name
            short collCode = getCollCodeFromName(dsDefICM, collName);
            // get list of entities
            String[] entities =
dsDefICM.listEntityNames(DKDatastoreDefICM.DK_ICM_USER_ITEM_TYPES);
            for (int i=0; i<entities.length; i++) {
                // check if prefix match
                if (entities[i].startsWith(entity)) {
                    DKItemTypeDefICM itemType = (DKItemTypeDefICM)
dsDefICM.retrieveEntity(entities[i]);
```

```

        System.out.println("Retrieved Item Type Definition of " +
entities[i]);
        dkCollection coll =
dsDefICM.retrieveItemTypeRelations(itemType.getId());
        dkIterator iter = coll.createIterator();
        while(iter.more()){
            ! ;
            DKItemTypeRelationDefICM itemTypeRel =
(DKItemTypeRelationDefICM) iter.next();
            System.out.println(itemTypeRel.getSourceItemTypeName() + " " +
itemTypeRel.getTargetItemTypeName() + " " +
itemTypeRel.getDefaultRMCode() + " " +
itemTypeRel.getDefaultCollCode());
            // Update if action is update
            if (action.equalsIgnoreCase("update")) {
                itemTypeRel.setDefaultCollCode(collCode);
                itemTypeRel.update();
                System.out.println("Update " +
itemTypeRel.getSourceItemTypeName() + " " +
itemTypeRel.getTargetItemTypeName() + " " +
itemTypeRel.getDefaultRMCode() + " " + collCode);
            }
        }
        &nbs! p;    }
    }
    // disconnect
    dsICM.disconnect();
    dsICM.destroy();
    System.out.println("Disconnected from datastore & destroying
reference.");
    } else {
        System.out.println("Usage: java ChangeGroup db user password entity
group action");
    }
}
// get collection id of collection name
public static short getCollCodeFromName(DKDatastoreDefICM dsDefICM, String
collName) throws Exception {
    DKDatastoreAdminICM dsAdmin = (DKDatastoreAdminICM)
dsDefICM.datastoreAdmin();
    DKConfigurationMgmtICM cfgMgmt = (DKConfigurationMgmtICM)
dsAdmin.configurationManagement();
    DKRMConfigurationMgmtICM rmCfg = cfgMgmt.rmConfigurationMgmt();
    dkCollection SMSDefinitions = rmCfg.listSMSCollections();
    &! nbsp; dkIterator iter = SMSDefinitions.createIterator();
    while(iter.more()){

```

```

        DKSMSCollectionDefICM smsCollDef =
(DKSMSCollectionDefICM)iter.next();
        if (collName.equalsIgnoreCase(smsCollDef.getName())) {
            System.out.println("CollCode "+
                smsCollDef.getName()+" ("+
                smsCollDef.getSMSCollectionCode() +")");
            return smsCollDef.getSMSCollectionCode();
        }
    }
    throw new Exception("Collection not found");
}
}

```

---





# Migration error codes and messages

This appendix provides the migration error codes and their corresponding messages.

## Migration error codes and messages

Example D-1 contains a list of error codes and messages you may encounter during migration.

*Example: D-1 z/OS migration error codes and messages*

---

ERRORMSG\_22000\_1 = Memory allocation failed

ERRORMSG\_22005\_1 = Failed to log on to CM V8 LS on database {2} as {3}.  
#EXAMPLE: Failed to log on to CM V8 LS on database icmnlbdb as icmadmin.

ERRORMSG\_22007\_1 = Checking user's privileges failed.

ERRORMSG\_22008\_1 = User does not have the needed {2}: {1}.  
#EXAMPLE: User does not have the needed Privilege Code: 1001.

ERRORMSG\_22040\_1 = Adding to mapping table {2} failed.

ERRORMSG\_22035\_1 = Definition of collection {3} on resource manager {4} failed.

---



## List and validate Content Manager definitions via SQL

This appendix contains sample SQLs that enable you to create a list of the definitions in your Content Manager system. This gives you an overview of how the internally used reference numbers match the names you use in the system administration client. In addition, we provide the SQLs that can help you spot any inconsistencies with the DB2 definitions in case the installation jobs were not completed successfully.

# Introductions

You may find the following sample SQLs useful to create information such as summary lists of the definitions of users, item types, and related DB2 tables based on the information from your Content Manager Library Server tables and some SYSIBM system tables. We also use the sample SQLs to validate table definitions.

We include SQLs that can do the following tasks:

- ▶ List users and their default settings.
- ▶ List item type definitions and DB2 table names.
- ▶ List access modules if you use CM V8.1 or V8.2.
- ▶ Validate Library Server and DB2 table definitions.
- ▶ Validate Library Server and DB2 view definitions.

You need to have SELECT privilege on the Content Manager V8.3 Library Server tables and the SYSIBM tables to use the SQLs in the following sections.

## List users and their default settings

The SQL shown in Example E-1 lists the users defined to the Library Server and the default values for some of their settings.

When using the SQL, remember to substitute the schema name of the Library Server tables, SQLID, with the one in your environment.

*Example: E-1 List Content Manager users and their current settings*

---

```
-- Set schema name of your Library Server tables
set current SQLID = 'ICM4BLS';

-- List user definitions
select distinct USR.UserId
, USR.UserPrivsetCode, UP.KeywordName UserPrivset
, USR.GrantPrivsetCode, UG.KeywordName GrantPrivset
, USR.DfltACLcode, ACL.KeywordName DfltACLcode
, USR.DomainID, DN.KeywordName DomainName
, USR.DfltRMcode, substr(RM.RMname,1,10) DfltRM

from ICMSTUsers USR

left join ICMSTNLSKeywords UP
on USR.UserPrivsetCode = UP.Keywordcode
and UP.KeywordClass = 11

left join ICMSTNLSKeywords UG
```

```

on USR.GrantPrivsetCode = UG.Keywordcode
and UG.KeywordClass = 11

left join ICMSTNLSKeywords ACL
on USR.DfltACLcode = ACL.Keywordcode
and ACL.KeywordClass = 13

left join ICMSTResourceMgr RM
on USR.DfltRMcode = RM.RMcode

left join ICMSTNLSKeywords DN
on USR.DomainID = DN.KeywordCode
and DN.KeywordClass = 17

order by USR.UserId
;

```

In Example E-2, we show a sample output from the SQL shown in Example E-1.

*Example: E-2 List of user definitions extracted from CM tables*

USERID	USERPRIVSET	GRANTPRIVSET	DFLTACLCODE	DOMAINNAME	DFLTRMCODE	DFLTRM
CCCHA	AllPrivs	AllPrivs	PublicReadACL	SuperDomain	1	ICM4BRM1
CCJAN	AllPrivs	AllPrivs	PublicReadACL	SuperDomain	1	ICM4BRM1
DEMO	AllPrivs	AllPrivs	DocRouteACL	SuperDomain	1	ICM4BRM1
ICMCO	UserDB2Connect	UserDB2Connect	PublicReadACL	DefaultDomain	1	ICM4BRM1
ICMLS	AllPrivs	ClientUserReadOnly	PublicReadACL	SuperDomain	1	ICM4BRM1

## List item type definitions and DB2 table names

When using the Content Manager system administration client to work with the item type definitions in Content Manager, it is not obvious which DB2 tables are used to store the metadata for a given item type or how the DB2 column names in these tables relate to the key fields assigned to the item type.

The SQL shown in Example E-3 creates a list of the item types defined in your Library Server. For each item type, the list shows the DB2 table name as well as the DB2 column names used for the attributes of the item type. The list should make Example E-1 a bit easier for you to see the relationship between the Content Manager definitions and the DB2 definitions.

*Example: E-3 List item types defined in Library Server*

```

--
-- Extract info from Content Manager tables

```

```

--
-- This SQL must be modified to match your local environment with
-- respect to:
-- NOTE 1. Database name of your Content Manager Library Server
-- NOTE 2. DB2 schema name for the Content Manager Library Server tables
-- NOTE 3. Remove references to SYSIBM.SysColumns table if you are
--         not authorized to access this DB2 system table
--
-- Find occurrences of the <*> marker below and change values
-- as described in the NOTES above.
--
-- To execute the SQL, save it in a file and run the DB2 command
-- from a DB2CMD window using the following parameters:
--
-- DB2 -t -fGetFieldNames.txt > FieldNames.txt
--
-- Where:
-- "GetFieldNames.sql" is the name of input file with the SQL
-- ">"                directs the output to a file
-- "FieldNames.txt"    is the name of the output file
-----
--
-- <*> NOTE 1: activate the following statement using the Content Manager
--         database name and a valid connect ID if you are not already connected
--         to the Library Server database
-- Connect to ICMNLSDB user <userid> using <password>;
--
--
-- <*> NOTE 2: Change ICM4BLS to the Library Server Schema name:
set current SQLID = 'ICM4BLS';

select Current SQLID Schema
, n.LanguageCode
, c.ComponentTypeID CompID
, substr('ICMUT' || right(digits(c.ComponentTypeID),5)
|| '001', 1, 14) DB2_TableName
, CASE
  when a.AttributeID > 999 THEN
    substr('ATTR' || right(digits(a.AttributeGroup),5) ||
right(digits(a.AttributeID),5), 1, 20)
  else substr(n.KeywordName, 1, 20)
  end DB2_ColumnName
, substr(n2.KeywordName, 1, 20) Itemtype
, CASE
  when a.AttributeID > 999 THEN
    substr(n.KeywordName, 1, 20)
  else ' '
  end CM_ColumnName
-- <*> NOTE 3: See note above on SYSIBM

```

```

, case when SysCol.Name IS NULL then 'ERROR' else 'OK' end Status

from
  ICMSTCompDefs c
left join
  ICMSTCompAttrs a
  on c.ComponentTypeID = a.ComponentTypeID
  and a.AttributeID > 1
left join
  ICMSTNLSKeywords n
  on n.KeywordCode = a.AttributeID
  and n.KeywordClass = 1
left join
  ICMSTNLSKeywords n2
  on n2.KeywordCode = c.ItemTypeID
  and n2.KeywordClass = 2
  and n2.LanguageCode = n.LanguageCode

-- <*> NOTE 3: see note on SYSIBM above
left join
  SYSIBM.SysColumns SysCol
  on SysCol.TBCreator = Current SQLID
  and SysCol.TBName = char('ICMUT'
                          || right(digits(c.ComponentTypeID),5)
                          || '001')
  and ( SysCol.Name = char('ATTR'
                          || right(digits(a.AttributeGroup),5)
                          || right(digits(a.AttributeID),5))
  or SysCol.Name = n.KeywordName )

-- <*> NOTE 4: set the value below to 999 if you only wish to list
-- user defined item types. Otherwise specify 0 to also include
-- the definition on system component tables
where c.ItemTypeID > 999

-- <*> NOTE 5: set the value below to 999 if you only wish to list
-- user defined attributes otherwise specify 0
and a.AttributeID > 999

order by n.LanguageCode, c.ItemTypeID,
        c.ComponentTypeID, a.AttributeID
;

```

---

The output created by the SQL in Example E-3 looks similar to the list shown in Example E-4. The list enables you to determine the DB2-related values such as table and column names based on Content Manager-defined terms such as item type name, and the reverse is also true.

The STATUS column is generated by linking the Content Manager information to the DB2 system catalog. In the case where a DB2 definition for an item type has failed (for example, running one of the ICMUJOBS failed), the status columns show “ERROR”. In this scenario, there is a mismatch between the Content Manager definitions and the available DB2 tables.

*Example: E-4 Item type and its related DB2 names for tables and columns*

SCHEMA	DB2_TABLENAME	DB2_COLUMNNAME	COMPID	ITEMTYPE	CM_COLUMNNAME	STATUS
ICM4BLS	ICMUT01000001	ATTR0000001000	1000	NOINDEX	SOURCE	OK
ICM4BLS	ICMUT01000001	ATTR0000001001	1000	NOINDEX	USER_ID	OK
ICM4BLS	ICMUT01000001	ATTR0000001002	1000	NOINDEX	TIMESTAMP	OK
ICM4BLS	ICMUT01002001	ATTR0000001003	1002	ICMSAVEDSEARCH	SSName	OK
ICM4BLS	ICMUT01002001	ATTR0000001004	1002	ICMSAVEDSEARCH	IsAttach	OK
ICM4BLS	ICMUT01002001	ATTR0000001005	1002	ICMSAVEDSEARCH	HasPar	OK
ICM4BLS	ICMUT01002001	ATTR0000001006	1002	ICMSAVEDSEARCH	Status	OK
ICM4BLS	ICMUT01002001	ATTR0000001007	1002	ICMSAVEDSEARCH	SelProcessName	OK
ICM4BLS	ICMUT01002001	ATTR0000001008	1002	ICMSAVEDSEARCH	SelLocationName	OK
ICM4BLS	ICMUT01002001	ATTR0000001009	1002	ICMSAVEDSEARCH	Versions	OK
ICM4BLS	ICMUT01002001	ATTR0000001010	1002	ICMSAVEDSEARCH	RetSemanticType	OK
ICM4BLS	ICMUT01002001	ATTR0000001011	1002	ICMSAVEDSEARCH	IsPublic	OK
ICM4BLS	ICMUT01002001	ATTR0000001012	1002	ICMSAVEDSEARCH	CompleteQuery	OK
ICM4BLS	ICMUT01003001	ATTR0000001013	1003	ICMSAVEDSEARCH	ViewID	OK
ICM4BLS	ICMUT01003001	ATTR0000001014	1003	ICMSAVEDSEARCH	ViewName	OK
ICM4BLS	ICMUT01003001	ATTR0000001015	1003	ICMSAVEDSEARCH	UsrSE	OK
ICM4BLS	ICMUT01003001	ATTR0000001016	1003	ICMSAVEDSEARCH	InternalSE	OK
ICM4BLS	ICMUT01003001	ATTR0000001017	1003	ICMSAVEDSEARCH	NumPars	OK
ICM4BLS	ICMUT01004001	ATTR0000001018	1004	ICMFORMS	ICMFORMNAME	OK
ICM4BLS	ICMUT01005001	ATTR0000001019	1005	ICMDRFOLDERS	ICMFOLDER_NAME	OK
ICM4BLS	ICMUT01005001	ATTR0000001020	1005	ICMDRFOLDERS	ICMFOLDER_DESC	OK
ICM4BLS	ICMUT01005001	ATTR0000001021	1005	ICMDRFOLDERS	ICMFOLDER_TIME	OK

From the sample output list in Example E-4, you can see information such as:

- ▶ The metadata for the NOINDEX item type is stored in DB2 table ICM4BLS.ICMUT001000001.
- ▶ The DB2 column name in DB2 table ICM4BLS.ICMUT01005001 where the value for Content Manager attribute ICMFOLDER\_NAME is ATTR0000001019.

## List access modules if you use CM V8.1 or V8.2

If you are running Content Manager Version 8.1 or Version 8.2, you may find the SQLs shown in Example E-5 useful. It provides you with a list of item types. For



each item type, the SQL provides the name of the access module that Content Manager uses to access the table.

**Note:** In Content Manager V8.3, these access modules are no longer used. Instead, Content Manager V8.3 uses dynamic SQL to access all item type-related tables.

*Example: E-5 List access modules for item types in CM V8.1 and CM V8.2*

---

```
- Set the schema name of the Library Server tables
set current sqlid = 'ITS08ADM';

-- List item type and status of the access modules
--
select current SQLID as Schema
, ItemTypeID, ComponentViewID, ComponentViewname
, AccessmoduleName
-- , AccessmoduleVers , PrevAccessmodule
, AccessmoduleStatus as AccStat
, Created
, case when SysPac.Creator IS NULL then 'ERROR' else 'OK' end Status
-- , Qualifier, Name
, COLLID
-- , OWNER, CREATOR
, Valid , Operative
-- , PDSname , Bindtime
from ICMSTCompViewdefs Viewdefs
left join SysIbm.SysPackage SysPac
on SysPac.Name = ViewDefs.AccessmoduleName
and SysPac.Qualifier = current SQLID
order by ItemTypeID, ComponentViewID
;
```

---

## Validate Library Server and DB2 table definitions

The SQL shown in Example E-6 can be used to validate the Library Server definitions for item types against the current DB2 definitions for the related tables.

One cause of Library Server definitions not being in synchronization with the actual DB2 definitions can be that one of the definition jobs created at installation time has not been run successfully.

In this case, the internal Library Server definitions do not match the current DB2 definitions. The output produced by the SQL will show any discrepancies between the Library Server and DB2 definitions.

*Example: E-6 SQL to validate LS item type definitions against DB2 table definitions*

---

```
--
-- Extract info from CM tables
--
-- This SQL must be modified to match your local environment with
-- respect to:
-- NOTE 1. Database name of your CM Library Server
-- NOTE 2. DB2 schema name for the CM Library Server tables
-- NOTE 3. Remove references to SYSIBM.SYSColumns table if you are
--         not authorized to access this DB2 system table
--
-- Find occurrences of the <*> marker below and change values
-- as described in the NOTES above.
--
-- To execute the SQL, save it in a file and run the DB2 command
-- from a DB2CMD window using the following parameters:
--
-- DB2 -t -fCheckTables.sql > TableStatus.txt
--
-- Where:
-- "CheckTables.sql"   is the name of input file with the SQL
-- ">"               directs the output to a file
-- "TableStatus.txt"  is the name of the output file
-----
--
-- <*> NOTE 1: activate the following statement using the CM database
--         name and a valid connect Id if you are not already connected
--         to the Library Server database
-- Connect to ICMNLSDB user <userid> using <password>;
--
-- <*> NOTE 2: Change ICM4BLS to the LS Schema name:
set current SQLID = 'ICM4BLS';

select
    Current SQLID as Schema
    , DB2_TableName
    , DB2_ColumnName
    , ComponentTypeID
    , AttributeID
-- , SysTab.Type , SysTab.Name , SysCol.Name
    , case when SysTab.Name IS NULL then 'ERROR' else 'OK' end TableStat
    , case when SysCol.Name IS NULL then 'ERROR' else 'OK' end ColStat
from
```

```

(select distinct
  substr('ICMUT' || right(digits(c.ComponentTypeID),5)
    || '001', 1, 14) as DB2_TableName
, CASE
  when a.AttributeID > 999 THEN
    substr('ATTR'
      || right(digits(a.AttributeGroup),5)
      || right(digits(a.AttributeID),5), 1, 20)
  else substr(n.KeywordName, 1, 20)
  end as DB2_ColumnName
, a.ComponentTypeID
, a.AttributeID
from
  ICMSTCompDefs c
left join
  ICMSTCompAttrs a
  on c.ComponentTypeID = a.ComponentTypeID
  and a.AttributeID > 1
left join
  ICMSTNLSKeywords n
  on n.KeywordCode = a.AttributeID
  and n.KeywordClass = 1)
as CMTAB

left join
  SYSIBM.SysTables SysTab
  on SysTab.Name = ucase(CMTAB.DB2_TableName)
  and SysTab.Creator = Current SQLID
-- and SysTab.Type = 'T'

left join
  SYSIBM.SysColumns SysCol
  on SysCol.TBCreator = Current SQLID
  and SysCol.TBName = ucase(CMTAB.DB2_TableName)
  and SysCol.Name = ucase(CMTAB.DB2_ColumnName)
where
-- <*> NOTE 3: set the value below to 999 if you only wish to list
-- user defined item types. Otherwise specify 0 to also include
-- the definition on system component tables
  ComponentTypeID > 0
order by
  ComponentTypeID
, AttributeID
;

```

Sample output from the SQL in Example E-6 is shown in Example E-7. Here the table definition for the NOINDEX item type is shown.

*Example: E-7 List of Library Server item type definitions and related DB2 table definitions and status*

SCHEMA	DB2_TABLENAME	DB2_COLUMNNAME	COMPONENTTYPEID	ATTRIBUTEID	TABLESTAT	COLSTAT
ICM4BLS	ICMUT01000001	COMPONENTID	1000	2	OK	OK
ICM4BLS	ICMUT01000001	ITEMID	1000	3	OK	OK
ICM4BLS	ICMUT01000001	VERSIONID	1000	4	OK	OK
ICM4BLS	ICMUT01000001	ACLCODE	1000	5	OK	OK
ICM4BLS	ICMUT01000001	SEMANTICTYPE	1000	6	OK	OK
ICM4BLS	ICMUT01000001	EXPIRATIONDATE	1000	10	OK	OK
ICM4BLS	ICMUT01000001	COMPKEY	1000	39	OK	OK
ICM4BLS	ICMUT01000001	CREATETS	1000	44	OK	OK
ICM4BLS	ICMUT01000001	CREATEUSERID	1000	45	OK	OK
ICM4BLS	ICMUT01000001	LASTCHANGEDTS	1000	46	OK	OK
ICM4BLS	ICMUT01000001	LASTCHANGEDUSERID	1000	47	OK	OK
ICM4BLS	ICMUT01000001	ATTR0000001000	1000	1000	OK	OK
ICM4BLS	ICMUT01000001	ATTR0000001001	1000	1001	OK	OK
ICM4BLS	ICMUT01000001	ATTR0000001002	1000	1002	OK	OK
...						

The output enables you to verify:

- Whether or not a DB2 table has been defined at all.

If the column TABLESTAT contains ERROR, then a DB2 table definition is missing.

- Whether or not the current DB2 table definition includes all of the expected columns.

If the column COLSTAT contains ERROR for one or more columns, you have most likely updated the item type definition in Library Server without running the associated batch job created by Library Server. Depending on which ICMMLSQn job type you choose at the time of installation, you may have to manually submit this job after updating the item type definitions. Also, review the job log for any of the ICMUJOBS you have run to discover potential errors.

## Validate Library Server and DB2 view definitions

The SQL shown in Example E-8 can be used to validate the Content Manager Library Server definitions for views against the current DB2 definitions.

One cause of Library Server definitions not being in synchronization with the actual DB2 definitions can be that one of the definition jobs created at installation time has not been run successfully.

In this case, the internal Library Server definitions do not match the current DB2 definitions. The output produced by the SQL will show any discrepancies between the Library Server and DB2 definitions.

*Example: E-8 SQL to validate view definitions in Library Server*

---

```
--
-- Extract info from Content Manager tables to validate
-- CM View definitions against actual DB2 definitions.
--
-- This SQL must be modified to match your local environment with
-- respect to:
-- NOTE 1. Database name of your CM Library Server
-- NOTE 2. DB2 schema name for the CM Library Server tables
-- NOTE 3. You need read access to SYSIBM.SysColumns and
--         SYSIBM.Systables to run this SQL
--
-- Find occurrences of the <*> marker below and change values
-- as described in the NOTES.
--
-- To execute the SQL, save it in a file and run the DB2 command
-- from a DB2CMD window using the following parameters:
--
-- DB2 -t -fCheckViews.sql > ViewStatus.txt
--
-- Where:
-- "CheckViews.sql" is the name of input file with the SQL
-- ">"           directs the output to a file
-- "ViewStatus.txt" is the name of the output file
-- -----
-- <*> NOTE 1: activate the following statement using the CM database
--             name and a valid connect Id if you are not already connected
--             to the Library Server database
-- Connect to ICMNLSDB user <userid> using <password>;
--
-- <*> NOTE 2: Change ICM4BLS to the LS Schema name:
set current SQLID = 'ICM4BLS';
--
-- Check availability of VIEWS
--
select
  Current SQLID Schema
, ViewDef.ComponentViewName
, substr(n.KeywordName, 1, 20) as KeywordName
, n.LanguageCode             as Lcode
, ViewDef.ComponentViewID     as CViewID
```

```

, ViewAttr.AttributeID
, case when SysTab.Name IS NULL then 'ERROR' else 'OK' end ViewStat
, case when SysCol.Name IS NULL then 'ERROR' else 'OK' end ColStat
from
  ICMSTCompViewDefs ViewDef
left join
  ICMSTCompViewAttrs ViewAttr
  on ViewDef.ComponentViewID = ViewAttr.ComponentViewID
  and ViewAttr.AttributeID > 1
left join
  ICMSTNLSKeywords n
  on n.KeywordCode = ViewAttr.AttributeID
  and n.KeywordClass = 1
left join
  SYSIBM.SysTables SysTab
  on SysTab.Name = ucase(ViewDef.ComponentViewName)
  and SysTab.Creator = Current SQLID
left join
  SYSIBM.SysColumns SysCol
  on SysCol.TBCreator = Current SQLID
  and SysCol.TBName = ucase(ViewDef.ComponentViewName)
  and SysCol.Name = ucase(n.KeywordName)
where
-- <*> NOTE 4: set the value below to 999 if you only wish to list
-- user defined item types. Otherwise specify 0 to also include
-- the definition on system component tables
ViewDef.ComponentViewID > 999
order by
  n.LanguageCode
, ViewDef.ComponentViewID
, ViewAttr.AttributeID
;

```

Sample output from the SQL in Example E-8 is shown in Example E-9. Here the view definition for the NOINDEX001 view is shown.

*Example: E-9 List of Library Server view definitions and DB2 status*

SCHEMA	COMPONENTVIEWNAME	KEYWORDNAME	LCODE	CVIEWID	ATTRIBUTEID	VIEWSTAT	COLSTAT
ICM4BLS	NOINDEX001	COMPONENTID	ENU	1000	2	OK	OK
ICM4BLS	NOINDEX001	ITEMID	ENU	1000	3	OK	OK
ICM4BLS	NOINDEX001	VERSIONID	ENU	1000	4	OK	OK
ICM4BLS	NOINDEX001	ACLCODE	ENU	1000	5	OK	OK
ICM4BLS	NOINDEX001	SEMANTICTYPE	ENU	1000	6	OK	OK
ICM4BLS	NOINDEX001	EXPIRATIONDATE	ENU	1000	10	OK	OK
ICM4BLS	NOINDEX001	COMPKEY	ENU	1000	39	OK	OK
ICM4BLS	NOINDEX001	CREATETS	ENU	1000	44	OK	OK
ICM4BLS	NOINDEX001	CREATEUSERID	ENU	1000	45	OK	OK

ICM4BLS	NOINDEX001	LASTCHANGEDTS	ENU	1000	46	OK	OK
ICM4BLS	NOINDEX001	LASTCHANGEDUSERID	ENU	1000	47	OK	OK
ICM4BLS	NOINDEX001	SOURCE	ENU	1000	1000	OK	OK
ICM4BLS	NOINDEX001	USER_ID	ENU	1000	1001	OK	OK
ICM4BLS	NOINDEX001	TIMESTAMP	ENU	1000	1002	OK	OK
....							

---

The output enables you to verify:

- Whether or not a DB2 view has been defined at all.

If the column VIEWSTAT contains ERROR, a view definition is missing.

- Whether or not the current DB2 view definition includes all of the expected columns.

If the column COLSTAT contains ERROR for one or more columns, you have most likely updated the item type definition in Library Server without running the associated batch job created by Library Server. Depending on which ICMMLSQn job type you chose at the time of installation, you may have to manually submit this job after updating the item type definitions. Also, review the job log for any of the ICMUJOBS you have run to discover potential errors.





# Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

## IBM Redbooks

For information on ordering these publications, see “How to get IBM Redbooks” on page 398. Note that some of the documents referenced here may be available in softcopy only.

- ▶ *Content Manager Implementation and Migration Cookbook*, SG24-7051
- ▶ *Content Manager V8.1 Migration Guide for Multiplatforms*, SG24-6877
- ▶ *Moving Data Across the DB2 Family*, SG24-6905
- ▶ *WebSphere Version 5 Application Development Handbook*, SG24-6993
- ▶ *eClient 101 Customization and Integration*, SG24-6964
- ▶ *Implementing Web Applications with Content Manager Information Integrator for Content and OnDemand Web Enablement Kit*, SG24-6338
- ▶ *WebSphere Portal Server and DB2 Information Integrator: A Synergistic Solution*, SG24-6433
- ▶ *Disaster Recovery with DB2 UDB for z/OS*, SG24-6370
- ▶ *Performance Tuning for Content Manager*, SG24-6949
- ▶ *IBM WebSphere Developer Technical Journal: Developing and Testing Message-driven Bean Applications with the MQ Simulator for Java Developers in WebSphere Studio V5.0*, SG24-6878
- ▶ *WebSphere MQ Integrator for z/OS V2.1 Implementation Guide*, SG24-6528

## Other publications

These publications are also relevant as further information sources:

- ▶ *IBM DB2 Content Manager for z/OS: Planning and Installing Your Content Management System*, GC18-7698
- ▶ *IBM DB2 Content Manager for z/OS: Migrating to DB2 Content Manager Version 8 for z/OS*, GC18-7699

- ▶ *IBM DB2 Content Manager Enterprise Edition V8.3: Planning and Installing Your Content Management System*, GC27-1332
- ▶ *IBM DB2 Content Manager for z/OS V8.3: System Administration Guide*, SC27-1335
- ▶ *IBM DB2 Content Manager Enterprise Edition V8.3: Migrating to DB2 Content Manager Version 8*, SC27-1343
- ▶ *IBM DB2 Content Manager Enterprise Edition V8.3: Application Programming Guide*, SC27-1347
- ▶ *IBM DB2 Content Manager Enterprise Edition V8.3: Application Programming Reference*, SC27-1348
- ▶ *IBM DB2 Content Manager for z/OS V8.3: Messages and Codes*, SC27-1349
- ▶ *IBM DB2 Content Manager Enterprise Edition V8.3: Installing, Configuring and Managing the eClient*, SC27-1350
- ▶ *IBM DB2 UDB V8 z/OS Utility Guide and Reference*, SC18-7427
- ▶ *IBM DB2 UDB V7 z/OS Utility Guide and Reference*, SC26-9945
- ▶ *IBM DB2 Universal Database Command Reference V8*, SC09-4828
- ▶ *IBM DB2 Universal Database Call Level Interface Guide and Reference, Volume 1*, SC09-4849
- ▶ *DB2 V7 for OS/390 and z/OS - Data Sharing: Planning and Administration*, SC26-9935
- ▶ *DB2 UDB for z/OS - Data Sharing: Planning and Administration*, SC18-7417
- ▶ *IBM Object Distribution Manager MVS/ESA: System Programmer's Guide*, SC34-3124
- ▶ *IBM Folder Application Facility MVS/ESA Application Programming Interface: System Programmer's Guide*, SC34-3120
- ▶ *IBM Folder Application Facility MVS/ESA Folder and Workflow Application: System Programmer's Guide*, SC34-3117
- ▶ *IBM DB2 Installation Guide*, GC26-9936
- ▶ *IBM WebSphere Application Server for z/OS V5.1, Getting started*, GA22-7957
- ▶ *WebSphere Application Server for z/OS V5.1: Applications*, SA22-7959
- ▶ *MQSeries Application Programming Guide*, SC33-0807
- ▶ *IBM DB2 Content Manager Enterprise Edition: Client for Windows Programming Reference*, SC27-1337
- ▶ *DB2 V7 z/OS Application Programming and SQL Guide*, SC26-9933
- ▶ *DB2 V7 z/OS SQL Reference*, SC26-9944

- ▶ *DB2 UDB V8 SQL Reference Volume 1*, SC09-4844
- ▶ *DB2 UDB V8 SQL Reference Volume 2*, SC09-4845
- ▶ *DB2 UDF V8 for z/OS Application Programming and SQL Guide*, SC18-7415
- ▶ *Program Directory for the Library Server for z/OS V08.03.00*
- ▶ *Program Directory for the Resource Manager for z/OS V08.03.00*
- ▶ *Program Directory for IBM DB2 Content Manager Toolkit for z/OS V8.3*
- ▶ *OS/390 and z/OS TCP/IP in the Parallel Sysplex Environment*, GM13-0026
- ▶ *IBM Communications Server - IP Configuration Guide*, SC31-8725
- ▶ *z/OS HTTP Server Planning, Installing, and Using*, SC34-4826

## Online resources

These Web sites are also relevant as further information sources:

- ▶ Web Enabling System/390 Applications Using WebSphere for OS/390, Java, and MQSeries:  
<http://www.redbooks.ibm.com/redpapers/pdfs/redp0027.pdf>
- ▶ IBM z/OS documentation:  
<http://www.ibm.com/servers/eserver/zseries/zos>
- ▶ WebSphere Application Server V5.1 for z/OS PDF files:  
<http://www-1.ibm.com/servers/eserver/zseries/zos/bkserv/zswpdf/was510.html>
- ▶ IBM DB2 for z/OS documentation:  
<http://www.ibm.com/software/data/db2/zos>
- ▶ Content Manager White paper *Performance Tuning Guide*:  
<http://www.ibm.com/support/docview.wss?uid=swg27003894>
- ▶ DB2 UDB for z/OS and OS/390 V7 publications:  
<http://www.ibm.com/software/data/db2/os390/v7books.html>
- ▶ DB2 UDB for z/OS and OS/390 V8 publications:  
<http://www.ibm.com/software/data/db2/zos/v8books.html>
- ▶ Technical information on Workload Manager configuration:  
<http://www.ibm.com/servers/eserver/zseries/zos/wlm/>

## How to get IBM Redbooks

You can search for, view, or download IBM Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy IBM Redbooks or CD-ROMs, at this Web site:

[ibm.com/redbooks](http://ibm.com/redbooks)

## Help from IBM

IBM Support and downloads

[ibm.com/support](http://ibm.com/support)

IBM Global Services

[ibm.com/services](http://ibm.com/services)

# Index

## Numerics

3270 green-screen interface 9

## A

access 335

access control 16, 18, 20, 32, 34

access control list (ACL) 29, 32–34, 118, 125, 198, 205–206, 208, 288, 299, 325

access module (pre CM V8.3)  
listing via SQL 386

### ACL

sample exit program 128  
user exit routine 125

### ACL code

extract information 120

ACL related functions 125

### ACL user exit

ICMACLPrivExit, parameters passed 129  
ICMACLPrivExit 125–126, 133, 138  
ICMACLPrivExit, activating 127  
ICMACLPrivExit, defining 126  
ICMACLPrivExit, dependencies 133  
ICMGenPrivExit 125, 132  
ICMGenPrivExit, activating 132

ACLs 111–112, 116, 126

action 35

predefined 35

action list 35

address space 155–156, 308, 310  
DB2 stored procedure 154

administration domain 18

Adobe Acrobat Internet plug-in 29

analyze

business requirement 14

annotation

semantic type 30  
viewing 48

APIs 18–19, 41, 45–51, 54, 58, 62–63, 68, 80, 82, 179–180, 182, 287, 355–356, 361, 372–373  
integration 80

log files 373

toolkit class path 361

application environment 155–159

WLM 155

Application Programming Interface (API) 9

asynchronous recovery process 42

attribute 21–24, 27–29, 31–32, 43

differences between Content Manager and ImagePlus 10

reference attribute 23

attribute group 21–22, 24

attribute update sample 60

auto-linking 22, 24–25  
enabling 24

Automatic Restart Manager (ARM) 308

availability 42

asystem administration client  
installation 172

## B

backup 42

batch job 118, 318, 321–322, 332–333

batch process 46, 62–63, 67  
input data 62–63

batch program

ICMPCACL 318

ICMXDFUR 318

business application 19, 36, 81, 179  
configuring user exit 95  
integrated line 179

business applications 232

business logic 62, 64, 75, 88, 138–139, 345

business requirement 14

## C

C programming language 132

Call Level Interface (CLI) 292–294

capacity planning 39, 42

casual scanning 9

CBRHTBSV 327, 330–331

CGI program 7, 11, 310–311, 327, 330, 333–334, 338

checklist 180, 185

migration from ImagePlus 180

checkout data

migrate 229

- child component 22–23, 56–57, 61
- CICS 7, 9, 226
- classification
  - document item 23
  - document part item classification 23
  - resource item 23
- CLASSPATH 47, 50, 67, 162, 173
  - environment variable 47
- CLI Driver 336
- Client for Windows 5, 18, 175
  - trace 285
- clients
  - traces and logs 285
- close method 75
- CMBCConnection connection 52, 65–66
- cmbicmenv.ini 305–306
- cmbicmsrvs.ini 305
- cmblogconfig.properties 294
- cmcfgls 159, 173–174, 198, 214–215
- COBOL 133
- code snippet 50
  - error handling 49
  - error handling with bean interface 49
  - SConnectDisconnectICM 50
  - TConnectionPool 52
- collection 39
  - control 147
  - Resource Manager, extract information 122
  - sample list 122
- collection name
  - change 146
- collection point 36, 81
  - configuring user exit 94
- collections
  - dealing 104
  - Resource Manager, dealing with 104
- component 15, 22–23, 32, 41
- component type
  - generating 226
- components
  - Content Manager 15
- comptypes 159, 173–174, 198, 214–215, 226
- configuration
  - server configuration utility 303–304
- configuration file 42, 160–161, 163, 170
- configure
  - Content Manager clients 303
  - document routing user exit 93
  - user exit in business application 95
  - user exit in collection point 94
- connect method 66
- connect user 115
- connection pool 52
- considerations
  - implementation 29
- considerations and planning aids 179
- container data
  - document routing 84
- containment relationship 24
- Content Manager 82, 223–228, 230, 279–281, 283, 287, 290–293, 296, 307–311, 313
  - ACL codes 120
  - CICS region 226
  - components 15, 19
  - database triggers 97
  - differences between z/OS and Multiplatforms 10
  - differences from ImagePlus 8, 10
  - environment, export 306
  - flexible data model 180
  - host APIs 45–48, 50–54, 57–60, 63–68, 71, 75
  - host integration 80–84
  - implementation 13–22, 24, 26–27, 29–31, 33–36, 38–43
  - import/sync user definitions 318
  - installation 153–155, 157–159, 161, 164, 169–173, 175
  - installation process 175
  - installation verification 175
  - installing 225
  - internal numeric values 118
  - introduction 3–11
  - item type definitions 383
  - load modules 302
  - migrate from multiplatforms 231–256, 259–260, 264, 269, 273
  - migration from ImagePlus 177–187, 189–192, 195, 197–198, 200, 203–204, 209, 213–214
  - multiple instances in same LPAR 295–300, 302–306
  - performance benchmarks 309
  - security and exits 111–113
  - security exits 97
  - system architecture 6
  - system architecture, z/OS 15
  - Toolkit for z/OS, configuring 163
  - Toolkit for z/OS, installation 161
  - upgrade 225

- very large implementation 104
  - vs ImagePlus IPFAF 9
- Content Manager clients
  - configure 303
- Content Manager for z/OS
  - installing 197
- Content Manager Toolkit 165–166
- Content Manager toolkit
  - traces and logs 294
- Content Manager Toolkit for z/OS 11, 16, 19, 41, 45–50, 54, 62–63, 68, 80, 82, 355–356, 361, 372
  - class path 361
  - features not available as compare to Multiplatforms 11
  - host integration 80
  - log files 373
- Content Manager V7
  - AVTnnnnn table 253
  - database 233–234
  - index class 246, 251
  - Library Server 236, 240, 246, 248, 251, 254
  - Library Server database 234
  - Object Server 235, 243, 249, 254–255
  - Object Server database 255
  - system 233, 236, 238, 246
  - table 237, 240, 245, 248–251
  - table name 245
  - terminology 246
- content support 8
- couple data set (CDS) 156

## D

- DASD 9, 181, 185–186
- data
  - deleting sample data 58
  - export 230
  - export Object Server data 255
- data model 4, 18–20, 22–23, 46, 48, 50, 54–55, 58, 62–63, 96
  - hierarchical structure 20
- data type 10
- database
  - catalog 239
  - create 233, 238
  - drop 158
  - Library Server (LS) 234
  - load data into z/OS 241
  - Resource Manager (RM) 234

- structure 234
  - unload Library Server database 240
- database index 31–32, 43
  - add 32
- database name 51, 59, 64, 68, 171–173, 233, 238–239, 241, 296–301, 305, 384
- dataset 63, 91–92, 124, 157, 160, 163, 175, 185–190, 198–201, 204, 206–207, 214–215, 220, 226, 229, 297, 299–300
- DB2
  - catalog 171–172
  - drop database 158
  - run-time client 16
  - stored procedures 5
  - subsystem 17
  - traces and logs 290
  - upgrade 225
- DB2 CommonStore for Exchange server 43
- DB2 CommonStore for Lotus® Domino 43
- DB2 CommonStore for SAP 44
- DB2 components
  - Library Server, grant access to 318
- DB2 database 5, 9–10, 16–17, 22–23, 115, 237, 242, 297–303, 305, 322
- DB2 Document Manager 44
- DB2 joblog 283
- DB2 libraries 162
- DB2 load utility 247
- DB2 object 157, 160, 228, 297, 299–301
- DB2 package 326, 328
- DB2 packages
  - Library Server 316, 323
- DB2 plan 298–299, 313, 315–316, 318, 320, 322, 330, 332–333, 335
  - Library Server 317
  - Resource Manager 327
- DB2 Records Manager 43
- DB2 stored procedure
  - address space 154
  - ICMLogon 112
- DB2 subsystem
  - multiple RM in same DB2 subsystem 300
- DB2 system 320–322, 325–329
  - Resource Manager and Library Server on same system 326
  - Resource Manager and Library Server on separate system 328
- DB2 table 31, 42, 139, 145, 154, 157–158, 160, 188, 191–192, 197, 220, 225, 235, 237, 240,

- 243–245, 248, 269, 290–291, 296, 298, 382–383, 386
  - default Library Server definitions 225
  - ImagePlus data 188
  - Resource Manager definitions 160
- DB2 table definition
  - extract 237
- DB2 table name
  - listing via SQL 383
- DB2 trace
  - Resource Manager 284
- DB2 trigger 138–139
  - calling a program from 139
  - define on root component table 140
  - validating key field values 138
- DB2 type 2 connection 166
- DB2 type 4 connection 165
- DB2 View 245
- DB2 view 325
  - additional background 324
  - Library Server table, access to 324
- DB2 z/OS
  - database 171
  - instance 311
- DB2DRDAT 291
- db2drdat 291
- DB2LOOK 237, 240
- DB2MOVE 240–241
- DB2TRC 292
- DBRM member 317, 326–328, 330
- DDCSTRC 291
- DDL 215, 217, 220–221
- decision point 37
- default collection 33, 39, 105–106, 117, 146–148, 375
  - changing, source code 375–376, 379
- default domain 123
- default value 281, 382
- DefaultDomain
  - predefined domain 41
- DEFTXT 116
- delete method 67
- deleting sample data 58
- deletion sample
  - sample
    - deletion 60
- development environment
  - z/OS 46
- DFSMS 179
- DGL0394A 335–336
- directory
  - create 229
- disk space 39
- displayError method 76
- Distributed Relational Database Architecture 291
- Distributed Relational Database Architecture (DR-DA) 291, 311
- DKDatastoreICM datastore 71–72, 74
- document 14, 18, 20, 22–30, 33, 36, 39, 42
  - differences between Content Manager and ImagePlus 10
  - import 175
  - non-operational 20
  - operational 20
  - semantic type 31
- document item
  - classification 23
- document part 23, 25–27
  - item classification 23
  - predefined 26
- document routing 34–35
  - container data 84
  - enhancement 8
  - exit function prototype 87
  - graphical builder 8
  - overload user exit 82
  - user exit 80
  - user exit configuring 93
  - user exit interface 85
  - user exit sample 82, 87
  - user exit, using 96
- document routing exit 81
- document routing sample 58
- domain
  - administration 18
  - create 41
  - extract information 122
  - management 40
- doPost method 75
- DRDA interface 166
- drop database 158
- DSNTRACE 337
- DSNUPROC 228
- dynamic SQL 128, 323–324, 387
- DYNAMICRULES 323



## E

- EAR file 76, 343–344, 347–348, 355–356, 362, 364, 372
  - create 345, 354
  - creating 76
  - purpose 344
- eClient 9, 18–20, 26–27, 29, 33, 35–37, 41, 80, 182
  - integration 108
- electronic archiving solution 44
- electronic data 8, 180
- encrypted password
  - copy from sample user 124
- entity listing sample 60
- environment variable
  - CLASSPATH 47
  - STEPLIB 47
- error
  - retrieve or import document 159
- error handling 49
- error message 76, 139, 143, 162, 175, 262, 286, 329–330, 335, 340, 345
- even
  - item 289
- event
  - system administration 288
- event log 289
- event logging 288
- example 106
- exception handling 49
- exit function prototype
  - document routing 87
- exit program 112–113, 127–129
  - ICMXLSLG 113
  - linking 137
  - make available to Resource Manager 146
  - sample 132
  - sample ACL exit program 128
- export
  - data 230
  - XML 306
- extract
  - DB2 table definition 237
- extract information
  - ACL code 120
  - domain 122
  - privilege set 119
  - Resource Manager 121
  - Resource Manager collection 122
  - user list from Content Manager 123

## F

- folder 24
  - differences from Content Manager to ImagePlus 10
  - semantic type 31
- folder creation sample 59
- folder manipulation sample 57
- Folder Workflow Application 9
- foreign key 21–23
- formatHeader method 76
- formatItem method 76
- formatResult method 76
- FRNPATRONS 234, 245

## G

- generating
  - system item and component types 226
- graphical builder
  - document routing 8
  - workflow process 37
- Greenwich Mean Time 8

## H

- history
  - semantic type 31
- host APIs 11, 16, 19, 41, 45–51, 54, 62–63, 68, 80, 82, 355–356, 361, 372
  - class path 361
  - features not available in z/OS 11
  - following features 11
  - host integration 80
  - log fiels 373
  - z/OS environment 11
- host integration
  - security exit 97
- host name 55, 101, 129, 172, 205, 298, 303, 305, 310, 344, 347–348, 357, 370, 372
- HTML code 352
  - test We servlet 68
- HTML file 70, 76, 344, 347, 351–352, 354
  - add to project 351
  - name 351
- HTML files
  - add to project 351
- HTTP 42
- HTTP procedure 146, 175
- HTTP request 19, 310, 344–345, 371
- HTTP Server 7

- executable module 199
- HTTP server 5, 11, 17, 55, 100, 103–104, 146, 154, 160–161, 199, 236, 262, 266, 296, 302, 309–311, 313, 327, 330, 333–335, 337–339, 345, 356, 370–371
  - CGI program 310
  - configuration 369
  - multiple instances 310
  - same procedure 302
  - several instances 313
  - single instance 311
  - trace 285
  - WAS configuration 371
- HttpServletRequest request 71–74

## I

- IBM Redbook 224, 313
- ICM path 51
- ICM program sample 53
- ICM sample program 53
  - compiling and executing 50
  - SConnectDisconnectICM 50
- ICMACLPrivExit 125–126, 133, 138
  - activating 127
  - defining 126
  - list dependencies 133
  - parameters passed 129
- ICMACLXT 112, 126
- ICMANNOTATION
  - document part 26
- ICMBASE
  - document part 26
- ICMBASESTREAM
  - document part 27
- ICMBASETEXT
  - document part 26
- ICMCONCT 320, 322, 335
- ICMCONCT user 115, 320, 322, 335–336
- ICMCONFIG 163
- icmconfig 164
- ICMCRYPT 164
- ICMGenPrivExit 125, 132
  - activating 132
- ICMGENXT 112, 132
- ICMLogon
  - DB2 stored procedure 112
- ICMMBIND 158, 298
- ICMMBKUP 115

- ICMMCACL 116, 158, 299, 318, 322
- ICMMCOMP 116
- ICMMDATA 115
- ICMMDFUR 116, 118–119, 124, 318, 322
  - input to job 118
- ICMMI75 250
- ICMMI76 250
- ICMMI7A 252
- ICMMIG2 204
- ICMMLSBD 158, 298
  - installation job 133
- ICMMLSCR 157, 297
  - installation job 112–113
- ICMMLSGT 158, 299, 318–319
- ICMMLS LD 158, 172, 225, 298, 321, 334
- ICMMLSQ1 158–159, 299–300
- ICMMLSQ2 157, 159, 299
- ICMMLSQ3 157, 159, 299
- ICMMLSQx 299–300
- ICMMMI70 227, 243–244
- ICMMMI71 227, 245
- ICMMMI72 227–228, 246–247
- ICMMMI73 227, 229, 248–249
- ICMMMI74 227, 249
- ICMMMI75 227, 229
- ICMMMI76 227, 230
- ICMMMI77 227–228, 230, 251
- ICMMMI7A 228, 248–249, 251
- ICMMMI7B 228, 230
- ICMMMI7C 228, 230
- ICMMMI7x 251
- ICMMMI80 247
- ICMMMI8B 228, 230
- ICMMMI8C 228, 230
- ICMMMI8x 251
- ICMMMIC0 187, 200, 214–215
- ICMMMIC1 192, 194–195, 214–215, 220
- ICMMMIC2 214, 220
- ICMMMIC3 214
- ICMMMIC4 187, 197, 214
- ICMMMIC6 214
- ICMMMIC7 214
- ICMMMIC8 214
- ICMMMIC9 187, 200, 214
- ICMMMICA 200, 214
- ICMMMICT 190, 193, 195, 215
- ICMMMICx 189–190, 195, 200, 219
- ICMMMIF2 202
- ICMMMIF6 202

- ICMMMIF7 203
- ICMMMIF8 189, 203
- ICMMMIF9 187, 200, 203
- ICMMMIFA 187, 200, 203
- ICMMMIFx 187, 189, 200
- ICMMMIP1 200, 204
- ICMMMIP2 206
- ICMMMIP3 207, 209
- ICMMMIPx 185, 189–190, 200–201
  - resulting output 185
- ICMMOPLN 327, 330, 335
- ICMMOSAP 327, 330–332, 335
- ICMMOSAR 327, 331–332, 335
- ICMMOSDI 327, 331–332, 335
- ICMMOSET 327, 330
- ICMMOSTT 327, 330
- ICMMRACF 112, 114
- ICMMRMBD 160, 215, 301, 325–327
- ICMMRMBR 301, 325, 328–329, 331
- ICMMRMCR 160, 300
- ICMMRMGR 332
- ICMMRMGT 160, 215, 302
- ICMMRMLD 160, 301
- ICMMRMSP 144
- ICMMRMWB 160–161, 333
- ICMMSORT 115
- ICMMUSTB 115
- ICMNOTELOG
  - document part 26
- ICMOBJT 207
- ICMPLSLG 112
- ICMPLSPC 133
- ICMPMIDR 228–229
- ICMRMCONTROL 145–146, 284
- ICMSERVERREPTYPE 322
- ICMSQL 158–159
- ICMSTCOLLNAME 214
- ICMSTNLSEKeywords 123
- ICMSTRESOURCEMGR 205, 214
- ICMSTSYSCONTROL 281
- ICMSTUSERS 125, 214, 322
  - Library Server table 113
- ICMSTUsers 247
- ICMSYADMEVENTS 288
- ICMUNTAR 164
- ICMVERIFY 164
- ICMXLSLG 112–115
- ImagePlus
  - background 9
  - content support 8
  - differences from Content Manager 8, 10
  - migrating 184
  - migration checklist 180
  - migration summary 200
  - ImagePlus DB2 table 187, 189
    - applicable data 187
  - ImagePlus Folder Application Facility (IPFAF) 9, 178–182, 184–186, 188, 193, 201–203, 208
  - ImagePlus for OS/390 178–179
  - ImagePlus IODM 9
    - vs Resource Manager 9
  - ImagePlus IPFAF 9
  - ImagePlus Object Distribution Manager (IODM) 9, 178, 180, 184, 188, 201, 205, 207, 211–212
  - ImagePlus OS/390 8, 177
  - implementation
    - considerations 29
    - large 104
  - import 175
    - document 175
    - user definition from RACF 115
  - import document
    - error 159
  - import job
    - provide input 124
  - import program 124
  - IMS 9
  - index 24, 31–32, 42–43
  - information center 54
  - Information Integrator for Content 11, 18, 34, 41, 80, 304, 349
    - integration 109
  - ingesting data 8
  - init method 75
  - input
    - to ICMMDFUR 118
  - input name 69
  - installation
    - accepting 200
  - installation job
    - ICMMLSD 133, 316
    - ICMMLSCR 112–113
    - ICMMRMBD 326–327
    - ICMMRMBR 328
    - ICMMRMGR 333
  - installation option 171
  - Installation process 170, 175, 315, 318
  - installation process 113, 254

- detailed description 254
- installing
  - Content Manager on z/OS 197
  - Resource Manager 199
- interface
  - non-visual bean 48
  - visual bean 48
- Invalid value 335
- IOCA 8
- IODM 179–180
- IP address 172, 239, 258, 262, 310–312
- IPFAF tablesets 181, 186
- IPFAF/API 9, 179–180
- IPFAF/FWA 9, 178–180
- IPFAF/WRAPI 178, 180, 182
- item 20–25, 27, 29–31, 33, 36–37, 138
- item creation sample 56, 59
- item deletion sample 56
- item event 289
- item modification sample 56
- item retrieval sample 56
- Item Type
  - properties screen 289
- Item type 289
- item type 10, 20–27, 29–32, 34, 39, 43, 181, 183, 187, 191–196, 206, 208, 215, 217–221
- item type definition
  - change 261
  - listing via SQL 383
- item type subset 23, 29
- items data
  - migrate 230
- IWPM client 9

## J

- Java Common Connectivity (JCC) 167
- Java connector 7
- Java files
  - add to project 351
- Java program 54, 62, 64, 67, 164
- Java source
  - add to project 352
- JCC 167
- JCL 62–63, 67, 92, 114, 156, 158, 160, 238, 245, 247–253
- JCL example
  - pre-link and link 92
- JCL sample 67

- JDBC class 162, 173
- JDBC classes from CLASSPATH (JCC) 162, 173
- JDBC driver 166
- JDBC trace 293
- join node 37
- JPEG image 10

## K

- KD APIs 58
- key field values
  - validating, DB2 trigger 138

## L

- large implementation 104
- LBOSDATA 234–235
- library
  - SICMSAM1 132
    - xxx.SICMINS1 library 115
    - xxx.SICMSAM1 128
    - xxx.SICMSAM1 library 113–114
- library files
  - download and import 348
- Library Server
  - ACL related functions 125
  - application environment 154
  - component definitions 254
  - configuration 339
  - Content Manager V7 system 233
  - database 181, 185, 190, 197, 214
  - database name 65
  - DB2 components 316, 318
  - DB2 packages 316, 323
  - DB2 plan 317
  - Enabling trace 280
  - features not available in z/OS 10
  - host name 305
  - installation 197
  - installation job ICMMLSBD 133
  - installation jobs 316
  - item information 312
  - multiple instances 296
  - new item 138
  - object indexing information 289
  - on same DB2 system as Resource Manager 326
  - packages and plans 298, 316
  - primary purpose 16
  - problems related to security 335

- Resource Manager on separate DB2 system 328
  - run ICMACC 200
  - same values 298–299
  - separate entities 24
  - separate parts 315
  - specified item type 65
  - trace level 282
  - traces and logs 280
  - V7 Object Server 254
  - WLM JCL procedure 158
  - Library Server (LS) 5, 7, 9–10, 16–19, 23–24, 38–39, 42, 50, 52, 55, 59–60, 64–66, 68, 75, 154, 157–158, 169–175, 179, 181, 185, 189–190, 192, 197, 199–200, 203, 206–207, 213, 217, 220, 224–225, 228, 230–231, 233–254, 256–259, 265, 274, 280–281, 287–289, 296–299, 301–302, 305, 308, 311, 315–317, 320–322, 324–330, 334–335, 337, 339–340, 372, 376, 382–384, 387
    - DB2 components, grant access to 318
    - definition update 260
    - installation 157
    - multiple, in same database 296
    - post installation 161
    - post installation tasks 159
    - shut down 226
  - Library Server programs
    - binding options 323
  - Library Server schema 317–318, 326, 328
  - Library Server table
    - DB2 view, access to 324
    - ICMSTUSERS 113
  - Line of business (LOB) 179
  - link 17, 19, 21–25, 29, 37, 39
  - link item sample 59
  - link listing sample 61
  - link manipulation sample 57
  - link removal sample 61
  - link type 21, 25
  - links data
    - migrate 230
  - load data 157, 160, 200, 220–221, 241, 248, 251–252
  - LOAD Library 302
  - load Library 42, 90, 92
  - load library 114, 146
  - log
    - Client for Windows 285
    - DB2 joblog 283
    - directory, single 8
    - event 288
    - single user ID, set to 8
    - system administration client 287
    - timestamp, standard 8
    - unique log ID 8
  - log file 174, 373
  - Logging 8
  - logging and tracing 8
  - logon
    - RACF 112
  - LPAR 295, 297–301
- ## M
- management class 235, 266–270, 272, 274
    - current definition 272
    - Migration Policy 268
    - migration policy 272
    - update 269
  - Maximum number 28, 68, 75, 310
  - maximum number
    - address space 310
  - media 17
  - metadata 5, 141–142, 179, 214, 311–312, 383, 386
  - migrate
    - checkout data 229
    - from Object Server to Resource Manager 254
    - items data 230
    - links data 230
    - parts data 229
    - user table 230
    - WIPITEMS data 229
  - migrating ImagePlus 184
  - migration 9
    - consider before migration 232
    - objects 231, 260
    - performing 200
    - planning and preparing 187
    - post migration activities 184, 214
    - post product 230
    - pre-migration steps 224
    - prerequisites 183–184
  - migration approach
    - defining 184
  - migration job 187–188, 191–193, 200, 215, 224, 226–227, 229, 231, 234, 242–243, 247–248, 252–253
    - customizing 227

- migration package
  - bind 228
- migration policy 39–40, 266–269, 272
- migration process 181, 185, 224, 233–236, 251–253, 256
  - complete detailed pictures 224
- migration requirements
  - accessing 184
- migration rule
  - define 266
- migration summary
  - ImagePlus 200
- migration table
  - create 228
- migration utility 179–181, 183–184, 187, 203, 217, 219–220, 225, 228–229, 235, 237, 248
  - DB2 packages 228
- migrator
  - setup 273
- MODCA 8, 10
- MODCA document 29
- modifying
  - display name for attributes 215
  - display name for item types 217
  - the attributes in an item type 219
- MQSeries workflow 35
- multiplatform environment 38, 40, 45, 48, 53, 80–81, 88
- Multiplatforms
  - Content Manager, differences from z/OS 10
- Multipurpose Internet Mail Extension (MIME) 23, 28–29
- MVT table 246, 251–253
  - additional jobs 246
  - member ICMMMI7C 252

## N

- next step 230
- NOINDEX CMDRFOLDERS 53
- non-operational document 20
- non-visual bean interface 48
- non-visual bean sample
  - compiling and executing 52
- note
  - semantic type 31

## O

- OAM 7, 9, 16–17, 38–39, 42, 98, 105, 146–147,

- 234, 236, 268
- OBJ\_ACTIONDATE 272–273
- Object Access Method (OAM) 5, 17, 179
- Object Server 226–229, 231, 233, 235, 244, 249, 254–256, 259
  - export data 255
  - exported data 255
  - migrate to Resource Manager 254
- Object Servers
  - map with Resource Managers 227
- object storage 38, 234
- object-level
  - access control 4
- objects
  - migration 231, 260
- onnect method 75
- operational document 20
- Oracle 16
- overload limit
  - work node 82

## P

- packages and plans
  - Resource Manager 325, 332
- parallel route 37
- Parallel Sysplex 308, 310, 312–313
  - effective use 308
  - main objective 308
  - multiple systems 313
  - other systems 308
- parts data
  - migrate 229
- password 124
  - encrypted, copy from sample user 124
  - invalid 339
- PATH 50, 162, 170, 173
- PDF file 10, 28
- PDS 83, 89, 92, 246–248
- PDS member 158–159
- performance 7, 22, 24, 31, 42–43
  - tuning 42
- performing
  - migration 200
  - post migration activities 214
- physical server 308
- plan name 301–302
- planning and preparing
  - migration 187

- plug-in
  - Adobe Acrobat Internet 29
- port number 55, 101, 310
- post migration activities 184, 214
- post product migration 230
- POSTCHSM 144–145
  - Resource Manager user exit 99
- POSTCHSMEXITPGM 145
- POSTDELT 144–145
- POSTDELTEXTITPGM 145
- post-installation job
  - ICMMLSLD, verify 225
- POSTPFCH 144, 146
- POSTPFCHEXITPGM 146
- POSTREPL 144–145
  - Resource Manager user exit 99
- POSTREPLEXITPGM 145
- POSTRTRV 144–145
  - Resource Manager user exit 99
- POSTRTRVEXITPGM 145
- POSTSTOR 144–145
  - Resource Manager user exit 99
- POSTSTORE 148
- POSTSTOREEXITPGM 145
- PRECHSM 144–145
  - Resource Manager user exit 99
- PRECHSMEXITPGM 145
- predefined domains 40
- PREDELT 144–145
- PREDELTEXTITPGM 145
- pre-fetch rule 98
- Pre-migration steps 224
- prepare method 75
- PREPFCH 144, 146
- PREPFCHEXITPGM 146
- PREQUSM 144–145
  - Resource Manager user exit 99
- PREQUSMEXITPGM 145
- PREREPL 144–145
  - Resource Manager user exit 99
- PREREPLEXITPGM 145
- prerequisites
  - migation 183–184
- PRERTRV 144–145
  - Resource Manager user exit 99
- PRERTRVEXITPGM 145
- PRESTOR 144–145
  - Resource Manager user exit 99
- PRESTORE 147

- PRESTOREEXITPGM 145
- privilege 33–34
- privilege group 33
- privilege set 32–34, 116–117
  - extract information 119
- privilege sets code
  - sample list of 120
- process
  - workflow 20, 34–37
- process method 67
- program
  - ICMMRACF 114
  - ICMXLSLG 115
- programs ICMXLSLG 114
- project
  - add HTML and Java files 351
- PublicDomain
  - predefined domain 40

## R

- RACF 33, 124
  - import user definition from 115
  - RACROUTE 114
    - user exit 114
- RACF logon 112
- RACF support 113
- RACF validation
  - user 114
- RACROUTE
  - RACF funtion 114
- recovery 42
- recovery process 42
  - asynchronous 42
- Redbooks Web site 398
  - Contact us xx
- reference attribute 21–23
- REPAIR 247
- replication 40
- resource item
  - classification 23
- Resource Manager
  - access data 334
  - access user definitions 258
  - current management classes 269
  - DB2 plan 327
  - DB2 trace 284
  - dealing with large collections 104
  - default collection 39

- Encryption key handler 326, 328
- existing objects 260
- extract information 121
- failover system 40
- features not available in z/OS 11
- grant specification, summary 334
- installation 199
- job grants execution privileges 160
- job loads default definitions 301
- Library Server definition 334
- Library Server on separate DB2 system 328
- load tables 256
- make exit program available 146
- map Object Servers 227
- migrate from Object Server to 254
- multipl, in same DB2 subsystem 300
- new collection 105
- new objects 259–260
- on same DB2 system as Library Server 326
- packages and plans 325, 332
- Plan name 301–302
- primary purpose 16
- problems related to security 337
- sample list of 121
- server definitions 257
- specific objects 289
- trace level 284
- traces and logs 283
- user exit 98, 103, 105
- user exit function prototype 100
- user exit interface 100
- user exit, example 100
- validate access to 262
- vs ImagePlus IODM 9
- Resource Manager (RM) 5, 7, 9, 11, 16–19, 23, 25–27, 33–34, 38–42, 55, 154, 158, 160, 170, 179, 183, 190, 199, 204–205, 207, 215, 225–227, 233–236, 244, 248–249, 254–269, 271–274, 280, 283–284, 287, 289, 295–296, 300, 308–311, 313, 315–316, 320, 325–335, 337–340
  - installation 160
  - multiple, in same subsystem 300
  - post installation 161
- Resource Manager collection
  - extract information 122
- Resource Manager exit
  - activating 145
  - parameters passed 144
  - user exit
    - Resource Manager 143
- Resource Manager package 160
- Resource Manager plan 160
- Resource Managers (RM) 5, 243, 245, 257, 261, 264–267, 300, 313
- resume list 36
- retention 9
- retrieve document
  - error 159
- return code
  - SQL, handle 143
- RMACCESS
  - table update 256, 258
- RMACCESS table 256, 258
  - current entries 258
  - query information 258
- RMSERVER
  - table update 256–257
- RMSERVER table 257, 259
- root component 22–23
- root component table
  - define DB2 trigger 140

**S**

- same DB2
  - database 298, 300–301, 303
- same LPAR 296
- sample
  - attribute update 60
  - document routing 58
  - document routing user exit 82, 87
  - entity listing 60
  - folder creation 59
  - folder manipulation 57
  - item creation 56, 59
  - item deletion 56
  - item modification 56
  - item retrieval 56
  - JCL 67
  - link item 59
  - link listing 61
  - link manipulation 57
  - link removal 61
  - non-visual bean, compiling and executing 52
  - transaction 56
- sample document 82, 87
- sample exit program 132
- sample output 121–123



- sample PL/I program
  - ACLPrivExit 134
- sample program 47, 51, 53, 55, 58–59, 64–65, 67, 129, 132, 141
  - compiling and running 50
  - main logic 65
  - output of SConnectDisconnectICM 51
  - output of TConnectionPool 53
- sample SQL
  - SQL sample 119
- SBTPATRONS 234
- Scan Notification Processing 9
- scanning 9
- schema name 114, 119, 173
- SConnectDisconnectICM
  - sample program 50
- SDSF log 157, 161
- SDSNEXIT 162
- SDSNLOAD 162
- SDSNLOAD2 162
- search
  - test searching 10
- search criterion 31, 64–65, 71, 75
- search method 66–67, 72, 75–76
- search result 57, 66, 72–73, 75, 107, 345
- search sample
  - sample
    - search 57, 61
- security
  - Library Server problems 335
  - Resource Manager problems 337
- security exit 112, 115, 126
  - host integration 97
- security validation 115
- select objservecode 227
- select property 280, 288
- select rmcode 227
- select statement 227, 290
- semantic type 23, 30–31
- server configuration utility 303–304
- servlet 41, 68, 71, 75–76, 108, 344, 354, 356, 363
  - define in project 353
- SICMINS1 115
- SICMMIN1 226
- SICMSAM1 library 132
- SICMSAMP 163
- simple trigger
  - example 139
- single-direction association 21, 23
- sizing 42
- SMP/E installation 154, 161, 163
- SNPA 9
- source code 76, 84, 87, 92, 97, 100, 106, 126, 132, 376
  - changing default collection 375–376, 379
  - search item type 71
  - values changing 55
- split node 37
- SQL 119–121, 128, 191, 198, 203, 208, 215, 217, 220, 234, 237, 240, 244–247, 250, 253–254, 257, 273–274, 381–387
  - extract information on Resource Manager 121
  - guide 128, 137
  - list DB2 table name 383
  - listing access module (pre CM V8.3) 386
  - listing item type definition 383
  - listing user and default setting 382
  - return code, handle 143
  - statement 125
- SQL statement 158–159, 323
- SQLJ driver 166
- start node 37
- STEPLIB 50, 113, 146, 162
  - environment variable 47
- stop node 37
- storage class
  - define new on remote destination 267
- storage subsystem 16–17, 38, 98
- stored procedure
  - address space 154
  - ICMLogon 112
- stored procedures
  - DB2 5
- stored procedures (SP) 316
- strategy
  - backup and recovery 42
- structure
  - database 234
- subprocess 37
- SuperDomain
  - predefined domain 40
- SYSIN DD 245, 247
- system
  - monitoring 42
- System administration
  - event 288, 290
  - event logging 289
- system administration 5, 9

- client 226, 280, 282–284, 287–289, 291, 293, 321–322, 334, 339
- event 288
- task 288
- user 246
- XML support 7
- System administration client
  - installation 161, 169, 171
- system administration client 5, 7–8, 18, 24, 29, 31–33, 40, 54, 81, 93, 117–118, 127, 132, 158, 161, 169, 172, 175, 198, 214–218, 236, 267, 273, 297, 303, 306, 381, 383
  - installation 169
  - installation screen 171
  - log 287
  - option 170
- system administration client connection 170
- system architecture
  - Content Manager for z/OS 15
- system definition
  - migrate 228
- system item type
  - generating 226
- system topology 15, 42

## T

- table definition 225
  - DB2, extract 237
- tableset 181, 186, 188–189
- TConnectionPool 52
- terminology 178
- text searching 10
- TIFF 8, 10
- Tivoli Storage Management (TSM) 5, 17, 38
- Tivoli Storage Manager (TSM) 7, 179
- toolkit
  - class path 361
  - Content Manager Toolkit for z/OS 19, 41, 45–50, 54, 62–63, 68, 80, 82, 355–356, 361, 372
  - integration, Content Manager Toolkit for z/OS 80
  - log files, Content Manager Toolkit for z/OS 373
- Toolkit for z/OS 7
  - configuring 163
  - installation 161
- trace
  - Client for Windows 285
  - clients 285

- HTTP server 285
- JDBC 293
  - level, Library Server 282
  - level, Resource Manager 284
- Library Server 280
- Resource Manager 283
- Resource Manager DB2 trace 284
- system administration client 287
- trace level
  - change 372
- TraceFlush 293
- tracelevel value 282–284
- TracePathName 293
- traces and logs
  - clients 285
  - Content Manager toolkit 294
  - Library Server 280
  - Resource Manager 283
  - UDB 290
- transaction sample 56
- troubleshooting 175
- TSM 7, 17, 39, 42, 98
- TSM application programming interface 7
- TSM collection 7
- type 2 connection
  - DB2 166
- type 4 connection
  - DB2 165

## U

- UDF 139
  - redefining 133
- UNIX System Service
  - output file 164
- Unix System Service 50–51, 53, 62–63, 161–162, 164
- UNIX System Services 7, 16, 19
- UNIX System Services (USS) 7
- UNLOAD1 201
- UNLOAD2 201
- UNLOAD5 201
- URL 311–313
- user
  - copy encrypted password from 124
  - ICMCONCT 115
  - user and default setting
    - listing via SQL 382
  - user default

- change 260
- User Defined Function (UDF) 126
- user definition 116
  - import from RACF 115
- user definitions
  - access 258
- user exit 18–19, 36, 80–83
  - compile and link, sample 89
  - configuring in business application 95
  - configuring in collection point 94
  - document routing 80
  - document routing, configure 93
  - document routing, sample 82, 87
  - document routing, using 96
  - ICMACLPrivExit 125–126, 133, 138
  - ICMACLPrivExit, activating 127
  - ICMACLPrivExit, defining 126
  - ICMACLPrivExit, dependencies 133
  - ICMACLPrivExit, parameters passed 129
  - ICMGenPrivExit 125, 132
  - ICMGenPrivExit, activating 132
  - overload document routing 82
  - predefined 98
  - RACF 114
  - Resource Manager 98, 103
  - Resource Manager, activating 145
  - Resource Manager, example 100
  - Windows client 107
- user exit function
  - Resource Manager, prototype 100
- user exit interface
  - document routing 85
  - Resource Manager 100
- user exit routine
  - ACL 125
- user exits 19
- user group 33
- User ID
  - RACF validation 114
- user ID
  - invalid 339
- user Id 226
- user IDs
  - compiled ACL table 158
- user import job
  - provide input 124
- user list
  - extract from Content Manager 123
- user table

- create 228
- migrate 230
- user tables definition
  - verify 230
- USERSTAT 202, 211
- uto-linking 24

## V

- version control 4
- version policies 27
- Versioning 23, 27–28
- virtual host 363, 365, 369, 372
  - define 369
- virtual node 37
- visual bean interface 48

## W

- Web application 19, 41, 76, 344, 349, 352, 354, 356, 370–371
  - Java environment 76
- Web interface 18
- Web page 71, 75, 344, 356
- Web project 76, 345–348, 351–352
  - create 345
  - define Web servlet 353
  - Java source code 352
  - web.xml file 76
- Web servlet 68, 71, 75, 344, 354
  - define in project 353
  - main body 71, 75
  - source code 71
- Web Site 347
- web.xml 344
- WebSphere
  - enterprise server configuration 358
- WebSphere application
  - create 362
  - log file, view 373
  - server 355–356, 358, 362, 372
- WebSphere Application Development Studio 343–344, 349
- WebSphere Application Server 7, 11, 17–19, 41, 77, 347, 354, 356
- WebSphere application server
  - configure 360
  - create 358
  - start 372
  - stop 372

- WebSphere configuration 357
- WebSphere enterprise application
  - configuration 362
- WebSphere integration
  - components of 356
  - prerequisite 356
  - purpose 356
- WebSphere server
  - plugin 369
  - plugin configuration 370
  - plugin, update 370
- Window client
  - user exit 107
- Windows client 7, 9, 18, 20, 26–27, 33, 35–37, 80, 182–183, 236, 297, 303, 321, 335, 339
  - error 335–336
  - error message window 335
  - integration 106
  - user exit 107
- WIPITEMS data
  - migrate 229
- WLM
  - application environment 298–299
  - procedure 298–299
- WLM application environment 155–156, 299
- WLM DNS 312
- work basket 34, 36, 81, 85
- work item 82–83
- work list 37
- work node 35–38, 80–82
  - different types 93
  - expiration time 82
  - overload events 82
  - overload limit 82
- work package 35–38, 58, 86
- workflow 4, 14, 18, 34–38
  - implementing 34
  - MQSeries 35
  - process 20, 34–37
  - resume list 36
  - work basket 34, 36, 81, 85
  - work item 82–83
  - work list 37
  - work node 35–38, 80–82
  - work package 35–38, 58, 86
- workflow process 36–37, 181
  - decision point 37
  - graphical builder 37
  - join node 37
  - split node 37
  - start node 37
  - stop node 37
  - subprocess 37
- Workload Manager 296, 308, 310–311, 313
- Workload Manager (WLM) 113–114, 127, 140, 154, 157, 236, 246
  - procedure, customize and copy 159

**X**

- XML 4
  - export 306
  - import data from 7
- XML support 7
- xxx.SICMINS1 library 115
- xxx.SICMSAM1 library 114, 128

**Z**

- z/OS
  - Content Manager, differences from Multiplatforms 10
- z/OS development environment 46
- z/OS partition 295
- z/OS system 308



## DB2 Content Manager for z/OS V8.3: Implementation, Installation, and Migration

(0.5" spine)  
0.475" <-> 0.875"  
250 <-> 459 pages







**Redbooks**

# DB2 Content Manager for z/OS V8.3

## Implementation, Installation, and Migration

**Toolkit, host  
integration, security,  
exits, and sample  
source code**

**Migration from  
ImagePlus, CM V2.3  
(OS/390), and CM V7  
(MP)**

**Advanced  
configuration topics**

This IBM Redbook covers IBM DB2 Content Manager for z/OS Version 8.3 implementation, installation, and migration. This IBM Redbook is aimed at architects, designers, developers, and system administrators of Content Manager systems.

We introduce Content Manager for z/OS V8.3 and provide information on how to implement a Content Manager for z/OS system. We cover general implementation topics and provide special z/OS related topics including Content Manager Toolkit for z/OS, host integration, Content Manager security, and Content Manager exits.

We also describe the product installation, and cover system migration from ImagePlus OS/390, Content Manager V2.3 for OS/390 (also known as Content Manager V7.1 for OS/390), and Content Manager V7 for Multiplatforms, to Content Manager for z/OS V8.3.

In addition, we cover information on traces and logs. We address advanced configuration topics, including configuring multiple Content Manager instances in the same LPAR, issues and considerations when setting up a Content Manager system in a sysplex environment, and changing default access to DB2 plans and packages. We discuss error situations you may encounter and how to analyze them.

By reading this redbook in conjunction with the product publications, we hope you learn the basics and gain insights on implementing, installing, or migrating a Content Manager system on z/OS platform. We also hope you like some of the hints and tips we provide throughout this redbook to help make your job easier.

**INTERNATIONAL  
TECHNICAL  
SUPPORT  
ORGANIZATION**

**BUILDING TECHNICAL  
INFORMATION BASED ON  
PRACTICAL EXPERIENCE**

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

**For more information:**  
[ibm.com/redbooks](http://ibm.com/redbooks)